Camera Positioning with CAD Model Matching

TOHOKU UNIVERSITY

Graduate School of Information Sciences

Camera Positioning with CAD Model Matching

A Dissertation Submitted for the Degree of
Doctor of Philosophy (Information Sciences)

Department of System Information Sciences

by

Chuantao ZANG

March 25, 2012

Camera Positioning with CAD Model Matching

Chuantao ZANG

Abstract

Computer vision and robotics have been widely used in industry to reduce costs and improve proficiency. In these robot vision applications, one important task is positioning/placing a camera attached on a robot to a desired pose so that the camera can get the required images in various applications, such as visual inspection, pick-and-place operation. In our study, we want to realize this task in a visual inspection system, where an inspection camera is mounted on a robot. The robot is controlled following a trained trajectory to move the camera close to each inspection region. However, when the inspection object is not put in the training pose, the trained robot trajectory will cause camera position error and this error will reduce the inspection efficiency greatly as the input images for inspection are not taken from the inspection regions.

To deal with this problem, it is necessary to place the camera to a new pose so that the camera can keep the same relative poses to the object to inspect it. Therefore, we are interested in developing effective algorithms for camera positioning. For a popular research topic in the computer vision and robotics, a large amount of approaches have been proposed in the literature. However, until now there are still many obstacles to accomplish this camera positioning task with various objects, because of the challenges from the diversity of the real objects; the large differences between the initial pose and the desired final pose which often cause local pose estimation algorithms to fail and the complexity of the scene which is partially occluded or heavily cluttered.

To deal with these challenges, our study aims for placing a camera attached on a robot relative to various kinds of objects even with partial occlusion or cluttered scenes at a high speed. This dissertation is organized as follows.

Chapter 1 describes our study objectives and motivations. A survey of the related approaches for camera positioning is presented in detail. From the survey of the state-of-art approaches, we found that no single approach can deal with this camera positioning task well in various cases. Consequently, it is necessary to combine several approaches for reliable system performances. We have proposed two combination strategies to place a camera with respect to planar objects and 3D objects respectively.

Chapter 2 presents our work on the GPU acceleration to address the need for faster homography solutions. Our contribution is that we have adopted GPUs to accelerate the Efficient Second-order Minimization (ESM) tracking algorithm to improve the whole system performances. With carefully parallelization and optimization for the ESM algorithm in CUDA, our GPU version ESM algorithm can work at about 20 to 30 times faster than its CPU version. The parallelization and optimization is described in detail so that this part can be a reference for those researchers in this filed. Besides the ESM algorithm, we extend the optimization work to the Thin-Plate Spline (TPS) deformable image registration. Evaluation experiments have been presented to verify the efficiency of our optimization techniques.

Chapter 3 describes our combination strategy for homography-based visual servo to place a camera with 2D planar objects for visual inspection. The combination is based on the Scale Invariant Feature Transform (SIFT) matching algorithm and the Efficient Second-order Minimization (ESM) tracking algorithm. In this combination strategy, the ESM tracking algorithm in chapter 2 provides fast homography estimation, meanwhile the initial homography, which is the key issue for the ESM tracking algorithm as a local optimization method, is provided by the SIFT matching algorithm. With this strategy, when the ESM tracking algorithm fails due to large initial error or occlusions, the homography from the SIFT matching algorithm is loaded to initializing the ESM tracking. By this combination strategy, the initial pose range has been enlarged and the system can continue moving the camera after occlusion happens. The efficiency of this strategy has been evaluated in the experiments and visual inspection applications.

Chapter 4 presents a coarse-to-fine strategy to estimate the object pose so as to move a camera relative to 3D objects for visual inspection. Initial object pose estimation is obtained by an efficient view-based pose estimation approach. Then the pose estimation is refined by local optimization approaches to provide a more accurate pose to move the camera. Finally, a homography-based visual servo scheme is adopted to reduce the influence from the error of system calibration and ensure the camera to be placed to its desired pose for visual inspection.

With this combination strategy, the initial pose is realized by a global search for the maximum similarity between the real image and those images rendered from Computer Graphics (CG image), hence the local optima as in other methods have be effectively avoided. To accelerate the search, we propose a lookup table method. Firstly, a series of image features are calculated from the image moments of all the CG images rendered from all possible poses in the search space. Then the principal component analysis (PCA) is carried out on these image features to reduce the feature dimension so as to reduce the size of the lookup table. With the PCA projected features a lookup table is created. By using PCA projection the features' distribution has been improved and the memory space for the lookup table has been effectively reduced. By searching the maximum similarity in a smaller space in the lookup table, a remarkable speedup of the pose estimation has been obtained. The real applications of visual inspection and pick-and-place operation of 3D objects based on this combination strategy are described in detail to evaluate the efficiency.

Chapter 5 concludes this dissertation and a discussion of future work is proposed.

Finally, the mathematical tools and robot kinematics commonly used in this dissertation are presented in the appendix parts.

Contents

List of Figures

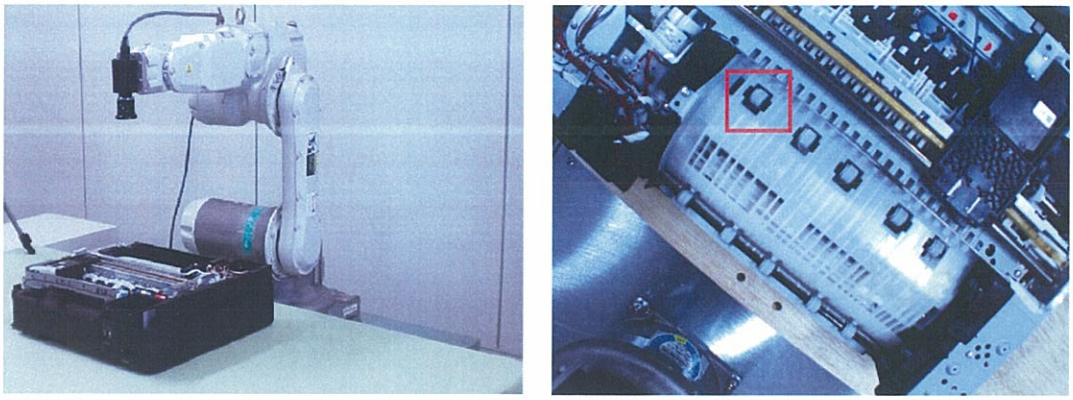
List of Tables

Chapter 1

Introduction

1.1 Motivation and Study Objectives

Computer vision and robotics have been widely used in industry and academia to reduce costs and improve proficiency. The applications range from tasks such as industrial machine vision systems, say, automated inspection system, robot guidance, to researches in artificial intelligence fields (augmented reality or virtual reality). In these robot vision applications, one important task is positioning/placing a camera attached on a robot to a desired pose so that the camera can get the required images for further applications, such as visual inspection, pick-and-place operation. In our study, we focus on realizing this task in a visual inspection system (Figure 1.1). In the system, an inspection camera is mounted on a robot. The robot is controlled following a trained trajectory to move the camera close to each inspection region. However, when the inspection object is not put at the training pose, the trained robot trajectory will cause camera position error and this error will reduce the inspection efficiency greatly as the input images for inspection are not taken from the inspection regions.

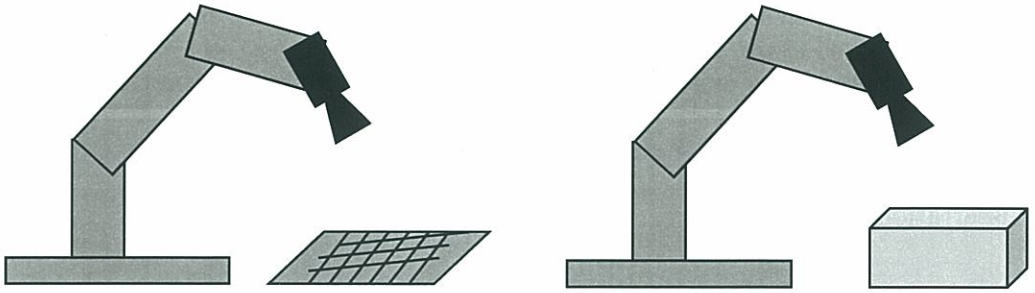


(a) (b)

Figure 1.1: A visual inspection system where we want to realize the camera positioning task. (a): The system is composed of a robot and an inspection camera. One of the inspection objects is a real printer. (b): An inspection region of the printer.

To deal with this problem, it is necessary to place the camera to a new pose so that the camera can keep the same relative poses to the object to inspect it. Therefore, we are interested in developing effective algorithms for camera positioning. For a popular research topic in computer vision and robotics, a large amount of approaches have been proposed in the literature (a survey in detail refers to section 1.2 on page 4). However, until now there are still many obstacles to accomplish this task with various objects, due to the several challenges as follows:

- The diversity of the real objects, such as planar objects, complex-structured objects, poor-textured objects.
- The large differences between the initial pose and the desired final pose which often cause local pose estimation algorithms to fail.
- The complexity of the scene which is partially occluded or cluttered where the conventional feature detection based methods perform poorly.
- The great computation demand of processing a large data array of an image which makes it difficult to realize fast system implementation.



(a) (b)

Figure 1.2: Two tasks of placing a camera attached on a robot to a desired pose relative to (a): a planar object and (b): a 3D object.

Among these challenges, dealing with various kinds of objects desires different techniques according to the different properties of objects. Ensuring a large initial pose range requires a robust algorithm capable of finding a global solution. Cluttered or partially occluded scenes make it much more desirable for an effective strategy to process them. Finally, high speed system implementation relies on the optimization of the algorithms to decrease the computational complexity, and the speedup due to the hardware acceleration, for instance, using such parallel processing hardware platform as field-programmable gate array (FPGA) or graphics processing unit (GPU) to accelerate the processing.

1.1.1 Study Objectives

To deal with above challenges, our study aims for placing a camera attached on a robot relative to planar and 3D objects (Figure 1.2) within a large initial pose range, even with partial occlusion or cluttered scenes. The processing task should be realized at a high processing speed.

The rest of this chapter is organized as follows. Section 1.2 describes the camera

positioning task in detail and lists the various related work on this task. Section 1.3 briefly introduces our implementation in each following chapter and the contributions of each chapter are shown in section 1.4. Section 1.5 lists the outline of this dissertation.

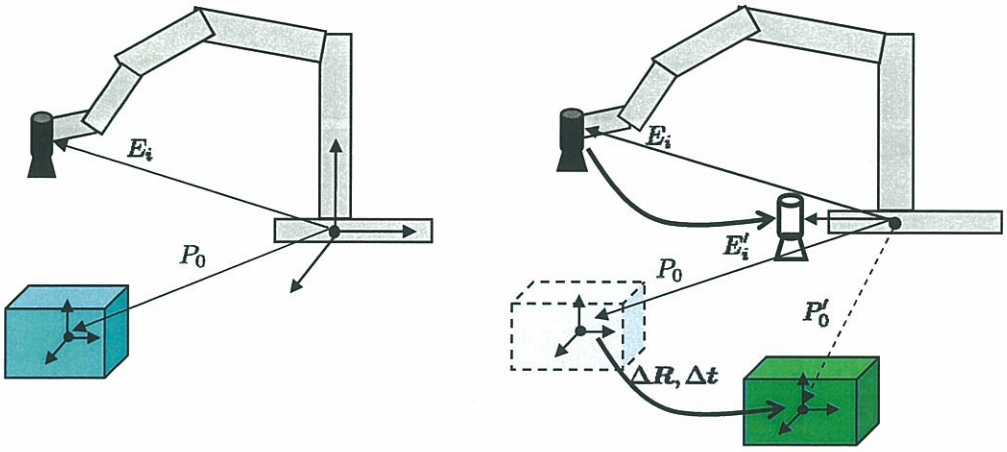
1.2 Related Work

This section describes the camera positioning task in detail and gives a survey of the related work on camera positioning with various objects. A summary of these approaches is given in the end of this section.

1.2.1 Problem Description

In this dissertation, we concentrate on realizing the camera positioning task in a visual inspection system (Figure 1.3). A series of N inspection operation is carried out on the objects. In the training/teaching phase (a), the object is put at the training pose P_0 and a set of N camera pose $\{E_i,\ i = 1, 2...N\}$ is manually chosen and saved from the robot joint angles. The word *pose* refers to the relation relationship between different coordinate frames. Then in the inspection phase (b), when the object is put at a different pose P_0', the trained camera trajectory is not applicable any more. In this case, a set of N new camera poses $\{E_i',\ i = 1, 2...N\}$ is required for the inspection task. Let $\Delta\boldsymbol{R}$ and $\Delta\boldsymbol{t}$ be the rotation and translation between the initial pose P_0' and training pose P_0' with the relationship

$$P_0' = \begin{bmatrix} \Delta\boldsymbol{R} & \Delta\boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} P_0, \tag{1.1}$$



(a) (b)

Figure 1.3: Camera positioning task in a visual inspection system. The training trajectory of pose $\{E_i,\ i = 1, 2...N\}$ is trained when the object is put at the pose P_0. The camera positioning task means finding a new trajectory of $\{E'_i,\ i = 1, 2...N\}$ for visual inspection when the object is put at different pose P'_0.

the new pose E'_i and training pose E_i need to keep the same relationship so as to inspect the object. That is,

$$E'_i = \begin{bmatrix} \Delta \boldsymbol{R} & \Delta \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} E_i,\ i = 1, 2...N. \tag{1.2}$$

Therefore, the problem consists in finding the transformation $\Delta \boldsymbol{R}$ and $\Delta \boldsymbol{t}$. The numerous approaches with this topic can be mainly classified into two categories.

- **Camera positioning by visual servo**. In the training phase, the object is put at pose P_0 (Figure 1.4 (a)). The robot is manually controlled to move the camera to pose P_A and capture one image I_A. Then when the object is put at an initial pose P'_0, the camera is first moved to P_A again and an image I_B is taken. Due to the pose difference between P_0 and P'_0, image I_B is also different from image I_A (Figure 1.4).

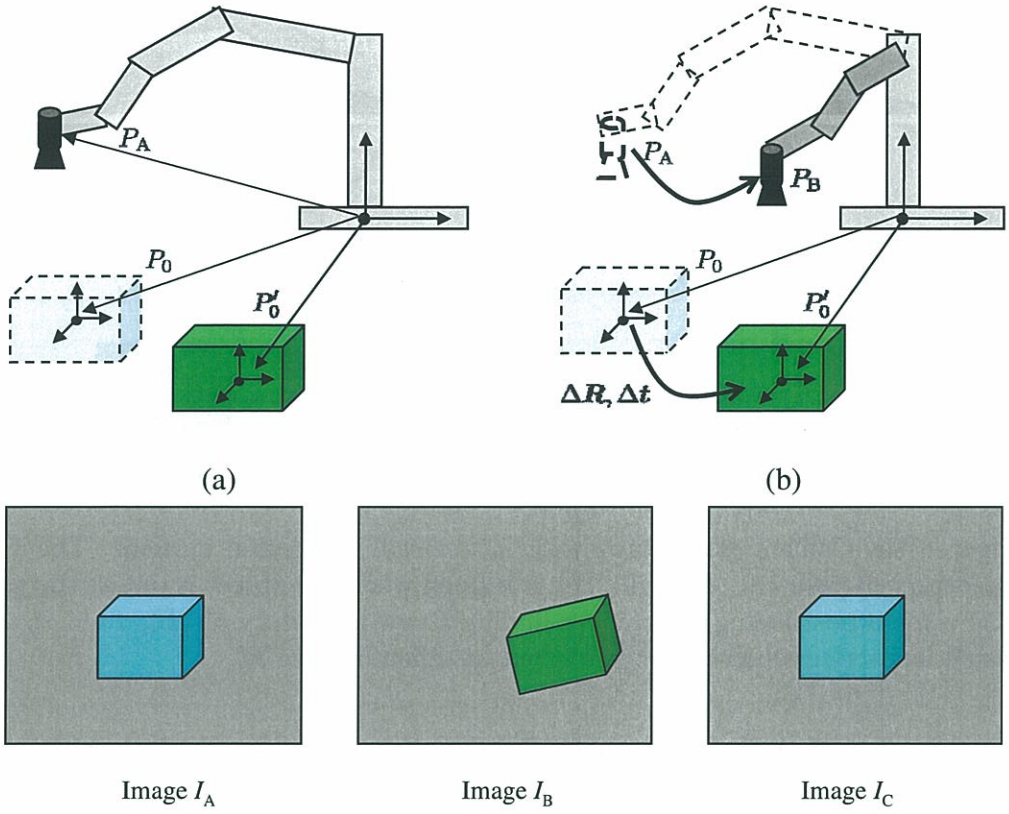


Figure 1.4: Camera positioning by visual servo. Image I_A and I_B are the images captured from P_A when the object is put at the training pose P_0 and unknown initial pose P_0' respectively. Image I_C is the image taken from camera's final pose P_B when the object is at its initial pose P_0'.

Then the image I_B is used as the initial image, image I_A as the reference image in a visual servo scheme. By minimizing the feature differences in the two images, the camera is finally moved to a pose P_B (Figure 1.4 (b)), where the camera can get an image I_C which is similar to the reference image I_A.

As the relationship between pose P_B and P_A keeps the same one as the object transformation:

$$P_B = \begin{bmatrix} \Delta \boldsymbol{R} & \Delta \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} P_A, \tag{1.3}$$

the needed transformation $\Delta \boldsymbol{R}$ and $\Delta \boldsymbol{t}$ for the new camera trajectory thus can be solved by

$$\begin{bmatrix} \Delta \boldsymbol{R} & \Delta \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} = P_{\mathrm{B}} P_{\mathrm{A}}^{-1}. \tag{1.4}$$

In this category visual information from the camera is used in the servo loop to move the camera to its desired pose (Figure 1.5 (a)). The image information is always used during the robot motion.

- **Camera positioning with an estimated pose**. In these strategies pose estimation algorithms are adopted to directly estimate the current object pose P_0'. Thereafter, with the relationship

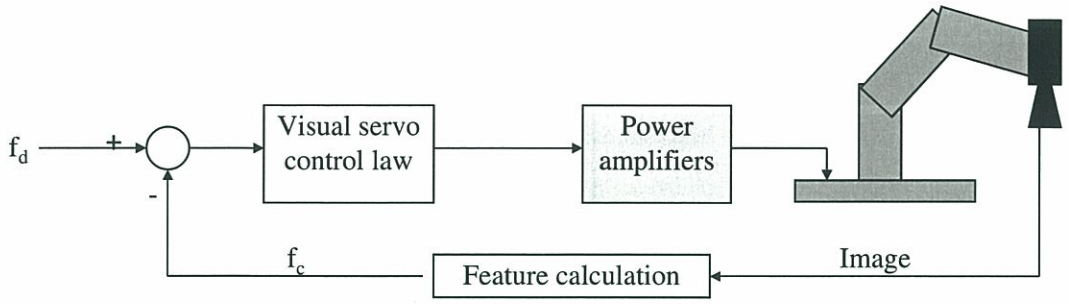
$$P_0' = \begin{bmatrix} \Delta \boldsymbol{R} & \Delta \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} P_0, \tag{1.5}$$

the transformation can be calculated from

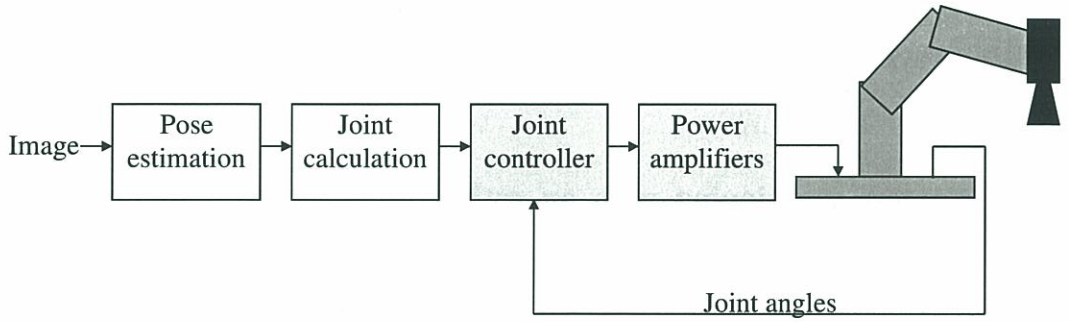
$$\begin{bmatrix} \Delta \boldsymbol{R} & \Delta \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} = P_0' P_0^{-1}. \tag{1.6}$$

Then the robot is moved directly to the desired pose with the estimated pose. During the robot motion, no image feedback or control is used (Figure 1.5 (b)).

An overall diagram is shown in Figure 1.6 on page 9. The following parts describe them in detail.



(a): Camera positioning by visual servo



(b): Camera positioning with pose estimation

Figure 1.5: Classification of the related work. (a): Image feedback is used in a visual servo scheme to place the camera. f_d and f_c are the desired features and extracted features from a current image respectively. (b): The camera is directly placed with joint information calculated from the pose estimation. The joint controller and power amplifiers are integrated into the robot control part.

1.2.2 Camera Positioning by Visual Servo

Visual servo refers to using computer vision data from one or more cameras to control the motion of a robot [1], [2]. This application task is treated as the regulation of a task function $\boldsymbol{e}$ which depends on the robot configuration and the time [3]. In the literature, visual servo approaches are mainly classified into three categories: Position-Based Visual Servo (PBVS), Image-Based Visual Servo (IBVS), Hybrid Visual Servo (Hybrid VS) [4], [5][1]. First we describe the related work of the two basic approaches: IBVS and PBVS, whose principles were proposed more than 20 years ago [6]. Then

[1]These two papers also give a survey of visual servo in detail.

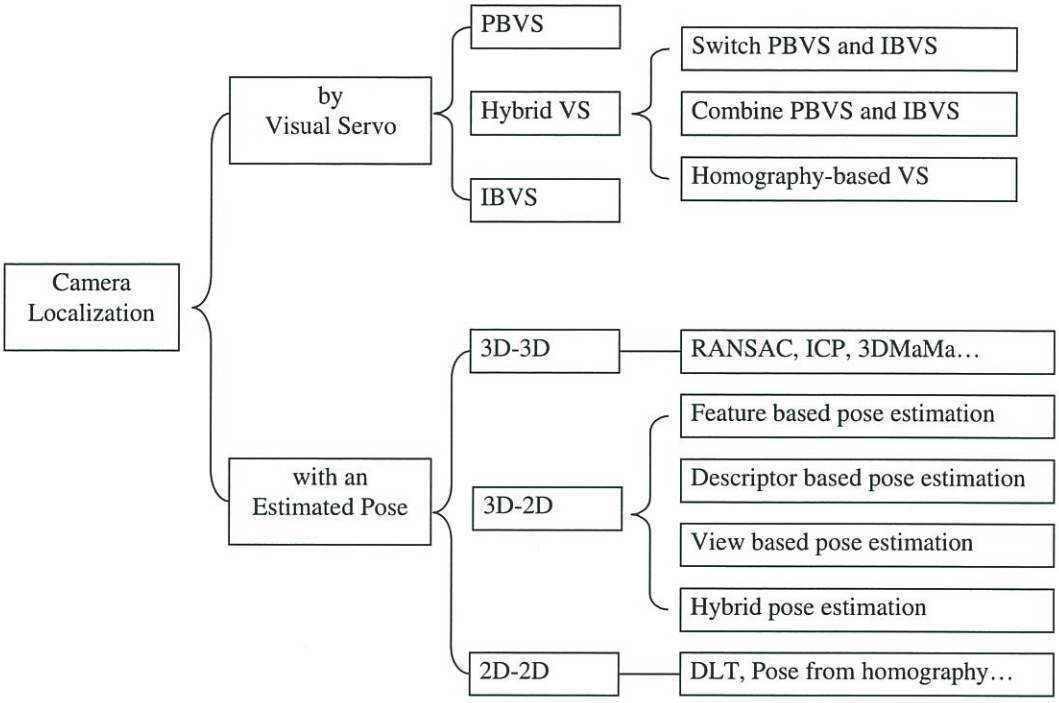


Figure 1.6: Overall diagram of the related work. The approaches are divided into two categories. Approaches in the visual servo category adopt the image information to control the camera motion, meanwhile the other category directly move the camera to its final pose with an estimated pose from the image, without any image feedback during the whole camera motion.

we present state-of-the-art approaches that have improved their performances.

(a) Position-based Visual Servo (PBVS)

The task function of PBVS is expressed in the 3D Cartesian space (Figure 1.7). The camera pose (translation and rotation relative to the object) is explicitly reconstructed from the obtained images and adopted as the features in the task function. Many solutions have been presented on PBVS in the literature, such as by Wilson [7], Ronen [8], Taylor [9], Thuilot [10], Malis and Chaumette [11].

In PBVS, the camera translation (up to a scale factor) and the camera rotation can be estimated from the Essential matrix [12], [13], [14], which can be recovered

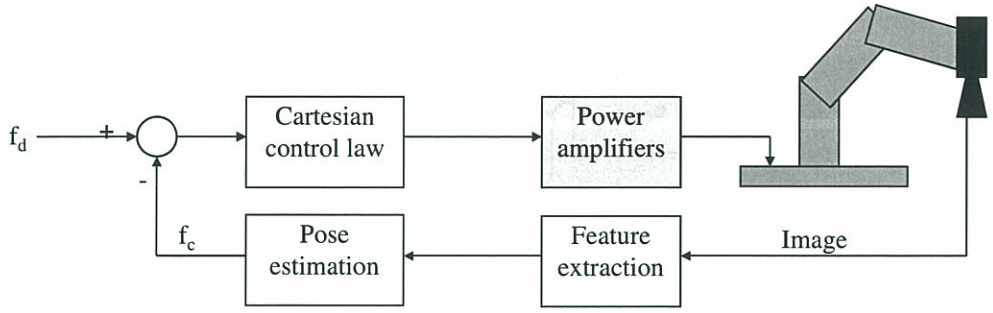


Figure 1.7: Diagram of position based visual servo (PBVS). The task function of PBVS is expressed in the 3D Cartesian space. The camera pose is explicitly reconstructed from the obtained images. f_d and f_c are the desired 3D pose and reconstructed pose from a current image respectively.

from a given set of matches between images measurements of two images by epipolar geometry [15]. With this Essential matrix, visual servo has been realized by Ronen [8], Rives [16].

However, to calculate the pose from a set of images, the camera intrinsic parameters and the 3D model of the object are usually necessary. Therefore, the system performance highly depends on the accuracy of the image measurement and the camera calibration, which is generally difficult due to image noises. Besides, the Essential matrix can not be estimated in those degenerated cases, for instance, when the target is planar or when the motion performed by the camera is pure rotation. For these reasons, it is better to estimate these parameters by using homography[2] matrix [18].

The PBVS scheme controls the robot in 3D Cartesian space and is globally asymptotically stable when the pose is assumed to be accurately estimated. Consequently, one drawback in position-based visual servo is that no control is performed in the

[2]Generally speaking, a homography is any transform H of n dimensional projective space $\mathbb{P}^n$ which is linear in projective coordinates (homogeneous coordinate) and invertible (thus it conserves globally the space with a fact we denote: $H(\mathbb{P}^n) = \mathbb{P}^n$). The geometric concept of homography in compute vision is a one-to-one correspondence between two sets of points, either in two Euclidean planes (Euclidean homography) or in two images (projective homography). More details refer to page 18 in [17] by Faugeras.

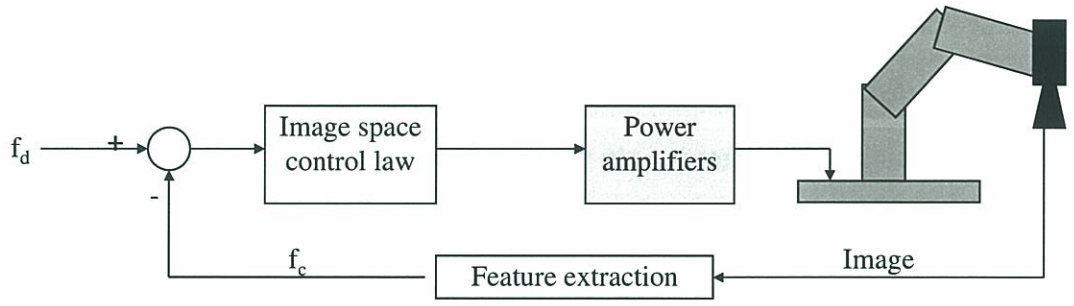


Figure 1.8: Diagram of image based visual servo (IBVS). The task function of IBVS is expressed in the 2D image space with features which can be directly derived from the obtained image. f_d and f_c are the desired 2D image features and extracted image features from a current image respectively.

image space, which implies that the object might not always be visible during the camera motion in the servo process (no doubt leading to the servo failure), especially in those cases where the initial robot position is far away from its desired one. To overcome this drawback, Chesi and Hashimoto have proposed a switching strategy among PBVS and backward motion for keeping features in the field of view [19].

(b) Image-based Visual Servo (IBVS)

Image based visual servo expresses its task function in the 2D image space with features which can be directly derived from the obtained image (Figure 1.8). For instance, image coordinates of points or the orientations of lines in an image. Traditional IBVS adopts the Cartesian coordinates of feature points in the error function to control the robot [4]. Similarly, cylindrical coordinates (ρ, θ) of features points are applied by Iwatsuki [20]. Applications with other geometrical primitives such as segments, lines, circles are given in [21],[22] and [23]. Attempts to use image moments in IBVS have been presented by Chaumette [24], Tahri [25], Fomena [26], Copot [27], Li [28], and so on. By selecting a suitable combination of image moments of a collection

of image points, various servo schemes have been realized.

As the task function in IBVS takes the image measurements into account, the control law makes the target remain visible in the field of view. However, for all those image measurement mentioned above, the depth estimation and an approximation of the interaction matrix are generally necessary. Consequently, several approaches with direct numerical estimation of the interaction matrix have been proposed in [29], [30], [31]. Without an analytical form of the interaction, these approaches have avoided the system modeling and system calibration. Deguchi has adopted the main eigenvalues of the principal component analysis to deal the extreme case where the interaction matrix is not available in analytical form [32].

Adopting control law in the 2D image space in IBVS ensures the object visibility during the servo process. However, image-based visual servo also suffers from several drawbacks [33]. The robot trajectory of robot in Cartesian space is unpredictable and robot might reach its joint limits in the workspace. The interaction matrix might become singular during the servo process, which results in an unstable system behavior (for instance, a sudden large instantaneous velocity in some extreme cases). Furthermore, features like image coordinates might induce local minima problem, which means that the final robot position does not correspond to the desired one, instead, the robot converges to another position.

(c) Hybrid Visual Servo

A series of hybrid visual servo approaches have been proposed to overcome the drawbacks in the classical PBVS and IBVS. By combining the advantages of classical visual servo techniques, these methods have improved the system performances on object visibility, local minima avoidance, joint limit avoidance, and so on.

To effectively combine classical visual servo techniques, Gans and Hutchinson proposed a switching strategy between IBVS and PBVS [34], [35]. In this strategy, a maximum error is defined for each scheme, which determines the system to switch to IBVS (or PBVS) if the 2D (or 3D) error norm exceeds its threshold. Hafez presented a hybrid control law with a 5D-objective function which allows determining a best weighting between IBVS and PBVS [36]. Then this weighting strategy is enhanced by an online boosting algorithm, in which the on-line boosting is adopted to produce a strong vision-based robot control starting from two weak algorithms: image-based and position-based visual servo algorithms [37]. By setting the error function of both classical algorithms separately in the image space and joint space, the robot is controlled with the constraint of the joint limit and the object visibility is kept. Kermorgant proposed a combination strategy of IBVS and PBVS [38]. By adding 2D image features in the PBVS, the object can be kept in the filed of view during the servo process.

To avoid the task singularity and local minima in IBVS, Malis has proposed a 2.5D visual servo scheme with a combination of 2D images features and 3D information [39], [40]. The feature vector $\boldsymbol{s}$, which is a combination of 2D features directly obtained from the image and 3D information estimated from the image, is selected as:

$$\boldsymbol{s} = [x, y, z, \theta\boldsymbol{u}^\top]^\top \tag{1.7}$$

where $x, y,$ and z are the metric coordinates of an image point, $\theta\boldsymbol{u}$ is the angle/axis representation[3] of the rotation. In this scheme, the rotation is decoupled from its translation part. The translation is controlled with a selected image point (center of

[3]Angle/axis representation of a rotation is introduced in section B in page 159.

gravity) while the rotation is derived from decomposition of the homography matrix.

To avoid the joint limits of a robot arm, Marey presented a new redundancy-based strategy [41], which is based on defining three functions for each joint: an activation function; an adaptive gain function; and a tuning function. These functions allow determining automatically the required sign and the suitable magnitude for the avoidance process at each joint.

Until now, all the schemes previously mentioned need the 3D parameters which need to be approximated or estimated from the image measurement. For instance, in IBVS the estimation of depth or the interaction matrix is needed. In PBVS, the 3D camera pose estimation is essential. As described in the PBVS part, the Essential matrix can not be estimated in those degenerated cases and estimation by using a homography matrix is preferred [18].

Homography-based Visual Servo approaches have been proposed in the literature. In these approaches, the homography between two images of the same planar objects is used to estimate the interframe displacements to move the camera. In these applications, some control strategies decompose the homography to explicitly reconstruct the camera motion (translation $\boldsymbol{t}$ and rotation $\boldsymbol{R}$), such as the applications by Faugeras [42], Vargas and Malis [43], [44]; others directly use the homography to control the robot without such decomposition, for instance, Fang [45] and Chen [46] to servo a wheeled mobile robot, Benhimane [47] and Silveira [48] for visual servo applications.

In contrast to these methods which need the 3D estimation, Benhimane and Malis have proposed a homography-based visual servo scheme [47], in which the homography is directly adopted in the control law. Consequently, no 3D parameter estimation is

needed.

To improve the performance of homography-base visual servo strategies for camera positioning, Deguchi adopts the decomposition of homography and Epipolar conditions to ensure the optimal trajectory of robot motion in the 3D space [49]. Hu proposed a unit quaternion-based controller to eliminate the singularity resulting from the traditional Euler-angle representation of the rotation matrix [50], [51]. The efficiency of quaternion-based controller in homography-based visual servo is also assessed by Koenig with simulation experiments [52]. Kumar has proposed robust homography estimation from multiple homographies induced by planar scene by employing geometric and subspace constraints [53]. With the homography estimation, a modified control law is adopted to compute the optimal camera trajectory.

1.2.3 Camera Positioning with an Estimated Pose

Camera positioning with estimated pose refers to placing a camera with direct pose estimation, no image information is used during the camera motion. The pose estimation deals with estimating the camera's position and orientation with respect to an object coordinate frame given a set of point correspondences. As an essential step in many computer vision applications, pose estimation has invoked a large amount of research works in the computer vision and robotics community. According to the dimension of the source data (a 2D image points set relative to a perspective projection or a 3D points set relative to an object frame), these pose estimation methods can be roughly classified into: 3D-3D, 2D-2D, and 2D-3D estimation approaches. Several earlier reviews of pose estimation approaches are discussed in [54] and [55].

(a) Pose Estimation from 3D-3D Correspondence

The 3D-3D estimation methods mainly cover the pose estimation between a 3D CAD point cloud $\boldsymbol{A} = \{\boldsymbol{a}_i = [x_i, y_i, z_i]^\top,\ i = 1, 2...N_\mathrm{A}\}$ and a retrieved 3D point cloud $\boldsymbol{B} = \{\boldsymbol{b}_i = [x'_i, y'_i, z'_i]^\top,\ i = 1, 2...N_\mathrm{B}\}$, where N_A and N_B are the number of points respectively. The retrieved 3D point cloud $\boldsymbol{B}$ can be obtained from a stereo camera system with epipolar geometry [56], [57], [58]; or from a structured-light sensor [59], [60], [61], [62]; or from a laser camera [63], [64], [65], [66], [67]. The estimation of the pose (composed of translation $\boldsymbol{t}$ and rotation $\boldsymbol{R}$) can be solved as a registration problem between could $\boldsymbol{A}$ and cloud $\boldsymbol{B}$

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \boldsymbol{t},\ (i = 1, 2...N_C), \tag{1.8}$$

where N_C is the number of corresponding pairs between $\boldsymbol{A}$ and $\boldsymbol{B}$. This problem can be solved with RANSAC algorithm [68], ICP algorithm [69], and 3DMaMa algorithm [70] and so on. The stereo vision system is easily set up but the accuracy of block matching algorithm to calculate the disparity decreases in the case of untextured objects. The pose estimation also suffers from the outliers of the 3D data which in fact have no corresponding points in the 3D CAD model.

(b) Pose Estimation from 2D-2D Correspondence

In these approaches, two 2D images are obtained either by observing the scene with two different cameras at the same time (stereo vision problem) or with a moving camera at two different instant (Structure from Motion problem) [17]. In both cases,

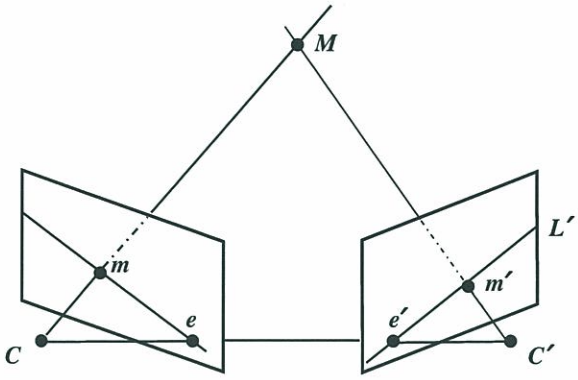


Figure 1.9: Epipolar geometry constraint. Given a projection m of a 3D point M in one image, the projection m' of M in the second image is restricted to lie on the epipolar line $L^{'}$ due to the constraint. Points e and e' are the epipoles. Points C and C' are the optical centers of the two cameras.

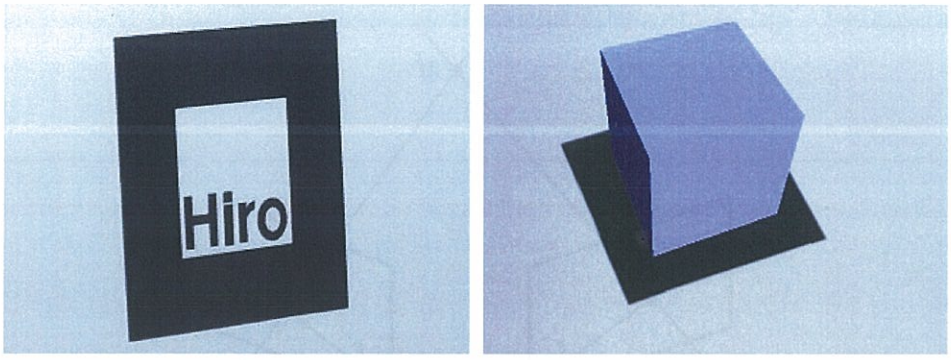
the epipolar geometry is the basic constraint (Figure 1.9) between two views.

Given a set of N known corresponding image points $\boldsymbol{m}_i \leftrightarrow \boldsymbol{m}_i'$ $(i = 1, 2...N)$ between two images or among multiple images, the pose can be estimated from the epipolar geometry and triangulation [71]. First a *fundamental matrix* $\boldsymbol{F}$ is estimated linearly from the known $N(N \geq 8)$ pairs of correspondence points $\boldsymbol{m}_i$ and $\boldsymbol{m}_i'$ in the two images with:

$$\boldsymbol{m}_i^{\top} \boldsymbol{F} \boldsymbol{m}_i' = 0 \tag{1.9}$$

Then an *essential matrix* $\boldsymbol{E}$ is derived from the fundamental matrix $\boldsymbol{F}$. The camera pose then can be calculated from this matrix $\boldsymbol{E}$ based on the Singularity Value Decomposition (SVD) method [17]. This classical formulation has been studied in the context of *structure from motion* problem in [72], [73], [74].

In real applications, finding the correspondence between two images often means a great challenge. Consequently, researches have been proposed to deal with unknown correspondence cases [75], [76]. Dallet adopts the Expectation Maximization (EM)



(a) (b)

Figure 1.10: ARToolKit. (a): One kind of planar marker in ARToolKit. (b): A virtual cuboid is overlaid on the planar marker with the estimated pose derived from plane homography.

algorithm to find the maximum likelihood estimation of the structure and motion [77]. Lin proposed a method to simultaneously compute the camera pose and point-to-point matching among image contours in [78]. An extension of their work of simultaneously estimating the camera pose and correspondence within a point set registration framework based on motion coherence is proposed in [79].

A special case is the correspondence between two images of a plane, i.e., homography relationship [17]. In 2D-2D pose estimation approaches, a reference template 2D image and a current 2D image are required. By estimating the interframe displacements from the image difference, the rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ can be obtained. Lucas and Kanade used the optical flow method to recover the translations in the consecutive images [80]. Malis proposed a better optimization algorithm- ESM algorithm to achieve a homography solution with a second order convergence but only with the same computational cost as other first order approximations [81]. his homography-based pose estimation is adopted in ARToolKit [82], a computer tracking library that allows for the creation of strong augmented reality applications. After the corner de-

tection in the planar marker, homography estimation is realized and further fast pose estimation is carried out (Figure 1.10). With this pose estimation, virtual imagery can be overlaid on the real scene.

(c) Pose Estimation from 3D-2D Correspondence

Pose estimation from 3D-2D correspondence consists of determining the camera pose with respect to an object coordinate system, based on the correspondence between a 3D object coordinate frame and a 2D image plane. As much information can be obtained from a CAD model, this kind of *model-to-image registration* approaches have shown great potential in various machine vision applications. A survey of related work can be found in [83]. These approaches are mainly classified into four categories: feature-based, local feature descriptor-based, view (appearance)-based, and hybrid approaches.

Feature-based Pose Estimation approaches try to solve the problem from the correspondence between the features in 3D CAD model and 2D images. Features can be point, line, free-form line, corner, edge, combination of features, and so on.

Using point features for pose estimation is a classical method and it is generally obtained using collinearity equations. Given a 3D point with coordinate $[x, y, z]^\top$ in the object coordinate frame and its corresponding 2D point with coordinate $[u, v]^\top$ in the image plane, their relationship is expressed with homogeneous coordinates as

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{Proj}_{3\times 4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{1.10}$$

where s is a scale and the matrix $\boldsymbol{Proj}$ is a projection matrix. Given $N(N \geq 6)$ corresponding pairs of points, a simplified form of this technique is the Direct Linear Transformation (DLT) algorithm [84]. Daniel proposed the Pose from Orthography and Scaling with ITerations (POSIT) algorithm to estimate the camera pose from at least four non-coplanar points with their corresponding 2D projections known in the image [85]. The focal length of the camera is assumed to be known. Its principle lies in approximating a perspective projection by a scaled orthographic projection, and iteratively adjusting the parameters of the scaled orthographic projection to converge to the real perspective projection.

A set of parallel lines in 3D space are adopted as the feature for finding the closed form solutions to estimate the camera pose [86]. First the distances from the optical center of the camera to these parallel lines are determined, then the pose parameters are recovered using the obtained distances. Ababsa presented a camera pose estimation approach based on 3D lines tracking [87]. An extended Kalman filter (EKF) method is used to incrementally update the camera pose in realtime. Li proposed a method of determining two pairs of vanishing points from a H-pattern of the car parking lot, which consists of two parallel lines and another line perpendicular to the two in a spherical image [88].

Ababsa proposed to combine a circular fiducials detection algorithm with a particle filter to calculate the camera pose [89]. With this framework the outliers due to either image noise or occlusion can be naturally discarded. Nagarajan adopted free-form lines to determine the camera pose using free-form lines [90]. Shi presented a new method for determining the camera pose from 2D to 3D corner correspondence [91]. Two cases of corners are discussed: the orthogonal corner and the general corner with

known space angles.

In edge-based approaches, a 3D CAD model is required to estimate the pose by aligning the edges in the projection of the CAD model to those edges extracted from a current image. This method was first proposed by Bouthemy [92]. Harris built the RAPiD system based on this algorithm by matching the projected CAD model edges to the those edges detected in a grayscale image [93]. Drummond and Cipolla have improved the RAPiD by adding the Iterative Re-weighted Least Squares (IRLS) with M estimator [94]. Comport and Marchand proposed an augmented reality framework with Virtual Visual Servo (VVS) approach [95]. These approaches are both computationally efficient and easy to realize. They are also stable for lightning changing and even for specular materials. The main limits of these methods are the jittering and the problem of background clutter with high textured objects, where the edges can not be detected correctly. Besides, as a kind of local optimization methods, these approaches can not find the right solution if the initial error is large, which means the solution is far from its initial point.

In real applications, automatic registration of a 3D object to 2D image is a difficult problem because it comprises two coupled problems: finding the correspondence and estimating the pose. This relationship belongs to the *chicken or the egg causality dilemma* in the means each can be solved if the other one has already been solved. A pose can be estimated from pairs of correspondences, while the correspondences can be obtained from a projective image with the known pose. Consequently, several approaches have been proposed to jointly estimate the pose and correspondence. David proposed the SoftPOSIT algorithm with point features [96]. This algorithm integrates an iterative pose approaches called POSIT [85] and an iterative correspon-

dence assignment technique called softassign [97] into a single iteration loop. An extension of this framework with line features is discussed in [98].

Local Feature Descriptor-based Methods estimate the pose by matching two sets of local feature descriptors (keypoints) in the reference image and a current image. A 3D metric model is produced off-line based on the reference images keypoints with their 3D coordinates. After matching them with the set of keypoints in the current image, the pose estimation is realized. Harris and Stephens adopted the corner points as the feature points [99]. Lowe proposed a Scale Invariant Feature Transform (SIFT) algorithm to estimate the 3D pose [100]. Compared with other features, the SIFT feature has been demonstrated to have a better performance with respect to variations in scale, rotation, and translation. However, its computation involves a high dimensional descriptor which is computational intensive and difficult to apply for realtime processing. To improve its performances, various algorithms have been proposed, including Affine SIFT [101], PCA-SIFT [102] and so on. Many parallelized algorithms have been realized on GPU or FPGA to accelerate SIFT for realtime performance [103]. An overview of these acceleration strategies are presented by Zhang in [104].

Partly inspired by the SIFT descriptor, the Speeded Up Robust Feature (SURF) by Herbert has shown comparable or better results to SIFT with only a fraction of the computational cost [105], [106].

The Binary Robust Independent Elementary Features (BRIEF) proposed by Calonder adopts binary strings as an efficient feature point descriptor [107]. As a result, BRIEF is very fast both to build and to match. Comparison between BRIEF and SURF shows that it yields a similar or better recognition performance, while run-

ning in a fraction of the time required by SURF. Several parallel processing hardware accelerated SURF descriptors are discussed in GPU-SURF [108], FPGA-SURF [109] and son on.

Another local descriptor DAISY presented by Tola is fast and efficient to compute [110]. It depends on histograms of gradients like SIFT but uses a Gaussian weighting and circularly symmetrical kernel. With this descriptor, an EM-based algorithm is adopted to compute dense depth and occlusion maps from wide-baseline image pairs. An improved version of DAISY descriptor O-DAISY feature has been realized with FGPA acceleration for wide baseline stereo system [111].

These local feature descriptor-based methods have shown good performances with textured objects. The limit is their heavy computational cost which makes them difficult for a realtime application even with the acceleration from various parallel hardware. And the most extreme cases with poor-textured objects, due to the lack of variable intensity information to build the descriptors, this kind of approaches can not ensure good performances.

View-based Pose Estimation Approaches compare the real image with pre-computed 2D views of the object to determine the camera pose [112], [113], [114], [115], [116]. These methods tried to search in the full six degree-of-freedom (6 DOF) geometric search space by clustering the views. Each image in the offline database is stored with the pose information where the image was taken. After finding the maximum intensity similarity between the search image and one image from the offline image database, the pose stored with this offline image is assumed to be the required camera pose. As the object texture (appearance) is directly used in the comparison between two images, these approaches can be classified into a texture (appearance)

based strategy.

Within these appearance-based approaches, no feature extraction step is needed. Therefore, the system implementation is simple and low cost. However, the limit of these strategy lies in the huge 2D view database size due to a search in the whole 6DOF space. Finding the maximum similarity in this database takes too much running time and often causes these approaches not suitable for real applications. Besides, the accuracy of pose estimation is limited to the resolution of the sampled poses in the offline database and often a subsequently pose refinement is needed. Ulrich proposed a hierarchical view-based approach to search the maximum similarly between the real scene and and an offline CG image database [117]. The pose refinement is realized with a least square method to ensure the estimation accuracy.

Hybrid Approaches are those combination strategies of above methods. As each single method above has its pros and cons, a combination of these methods makes the system more effective. Ababsa proposed a new robust camera pose estimation algorithm based on 3D model tracking [118]. The point and line features are combined in order to handle partial occlusion and increase the accuracy. Masson proposed a combination of texture and contour based visual tracking approach [119] but it is limited to planar objects. Pressigout and Marchand realized a hybrid tracking approach by merging the edge features and texture features in the object function [120]. Ladikos combined the template-based and feature-based methods [121]. The extended ESM algorithm is used as the default tracking method while the Harris Corner feature points are used to deal with the larger interframe displacement. Choi and Christensen proposed a combination strategy of edge and keypoints [122]. The SURF keypoints is used to estimate the initial pose in Global Pose Estimation (GPE)

with RANSAC method, then an edge-based tracking scheme is utilized in the Local Pose Estimation (LPE). These combinations show a better performance than a single method. But when considering those textureless objects, above combination methods perform poorly due to the feature extraction error or even fail to work.

A combination approach of stereo vision and model-based tracking to three-dimensional vehicle pose estimation is proposed by Barrois [123]. After computation of stereo and optical flow on the investigated scene, a four-dimensional clustering approach separates the static from the moving objects in the scene. The iterative closest point algorithm (ICP) is adopted to estimate the vehicle pose using a cuboid as a weak vehicle model.

A multi-flash-camera (MFC) based pose estimation with poor textured object (such as an industrial part) or heavy clutter scene is proposed by Liu [124]. A multi-flash-cameraprovides accurate separation of depth edges and texture edges in such scenes. Then the pose is estimated by a search for the best match between two kinds of depth edges: one kind is obtained from the MFC images and the other kind is computed offline using 3D CAD model of the objects.

Combination of the polyhedral 3D model and 2D surface texture is adopted to to simultaneously estimate 3D pose and camera zoom parameters from sequential images [125]. The 3D pose parameters and camera focal lengths, which yield the best match between the current image and the reference image, are estimated precisely using gradient descent optimization.

1.2.4 Summary of the Survey

We have listed the related work on the camera positioning above. The visual servo scheme can effectively place the camera to its desired pose. But usually the 3D parameters (such as the depth information) need to be approximated or estimated from the image measurements in these schemes. This kind of depth estimation has limited the application of visual servo with complex 3D objects, as it is usually difficult to reconstruct accurate depth information from the images only, due to the depth information loss in the 3D-2D image formation process. However, the particular homography-based visual servo scheme can avoid the depth estimation limit when it is carried out with planar objects. Therefore, this scheme has be adopted in our system for planar objects.

For 3D objects, the CAD model is used to reconstruct the pose information of the objects. A simple idea is that we can reconstruct the 3D parameters from the pose and adopt them into a position based visual servo to control the camera in 3D space. However, due to the calibration error in the camera parameters, the reconstruction can not always provide high accuracy 3D parameters. Consequently, in our system we directly adopt the pose information to move the camera so that we can avoid the errors from the reconstruction process.

From the previous survey of the state-of-art approaches, we found that no single approach can deal with this camera positioning task well in various cases. Consequently, adopting several combination approaches is necessary to obtain reliable system performances.

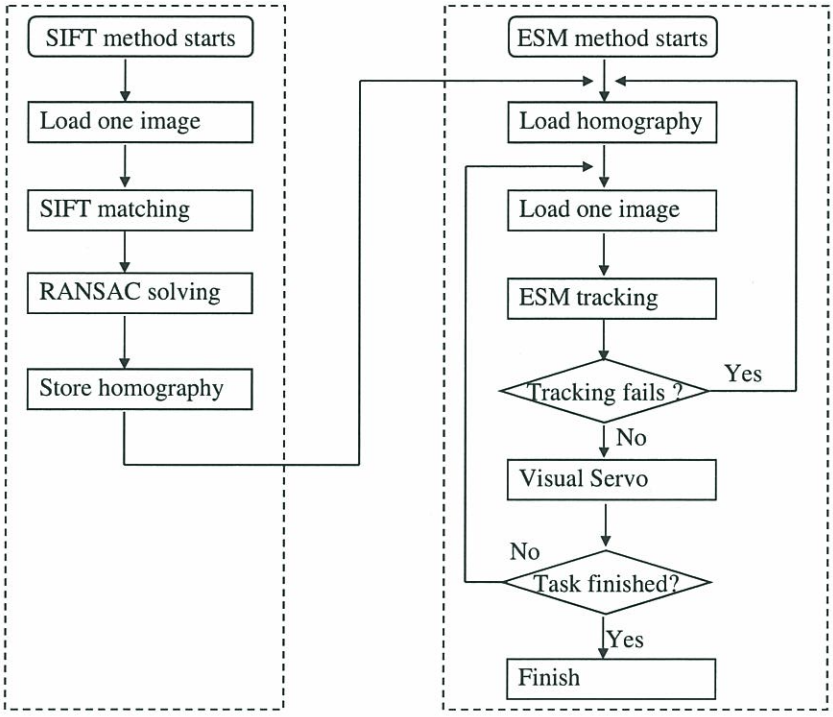
1.3 Our Implementation of Camera Positioning in a Visual Inspection System

This section briefly introduces our combination strategies to place a camera with respect to planar objects and 3D objects respectively to realize the visual inspection task in this dissertation. More detail refers to the following chapters.

1.3.1 Camera Positioning with Planar or 3D objects

For planar objects, we adopt the homography-based visual servo scheme to place a camera to its desired pose. The process flowchart is shown in Figure 1.11. The combination strategy for homography estimation is based on the Scale Invariant Feature Transform (SIFT) matching algorithm and the Efficient Second-order Minimization (ESM) tracking algorithm. The SIFT matching algorithm calculates the homography estimation and stores it into the shared memory. The ESM tracking algorithm provides fast homography estimation for the homography-based visual servo scheme to realize the camera positioning task. In the beginning, the ESM tracking algorithm adopts the homography from the SIFT method as the initial solution to start the tracking. When the ESM tracking fails due to occlusion, the SIFT method's homography is loaded into the ESM tracking method again to restart the ESM tracking algorithm. Both methods run simultaneously in the system. With this combination strategy, the system can realize fast and robust homography estimation to move the camera to its desired pose.

For 3D objects, a coarse-to-fine combination strategy is adopted to place the camera. The process flowchart is shown in Figure 1.13. Initial pose estimation is



(a) SIFT method (b) ESM method

Figure 1.11: Process flowchart of the combination strategy adopted in placing a camera with respect to 2D planar objects. The ESM tracking algorithm provides fast homography estimation for a homography-based visual servo scheme to realize the camera positioning task. The SIFT matching algorithm provides the homography solutions to initialize the ESM tracking algorithm in the beginning and other cases where the ESM tracking fails due to occlusion. Both methods run simultaneously in the system.

obtained by a global search method in the coarse step. Then the pose estimation is refined by local optimization approaches in the fine step to provide a more accurate pose to move the camera. Finally, a visual servo scheme is adopted to reduce the error from system calibration and ensure the camera to be placed to its desired pose.

1.3.2 Implementation of Visual Inspection

Our implementation of visual inspection is implemented by 2D appearance matching. The inspection task is realized by a template matching approach (Figure 4.17)

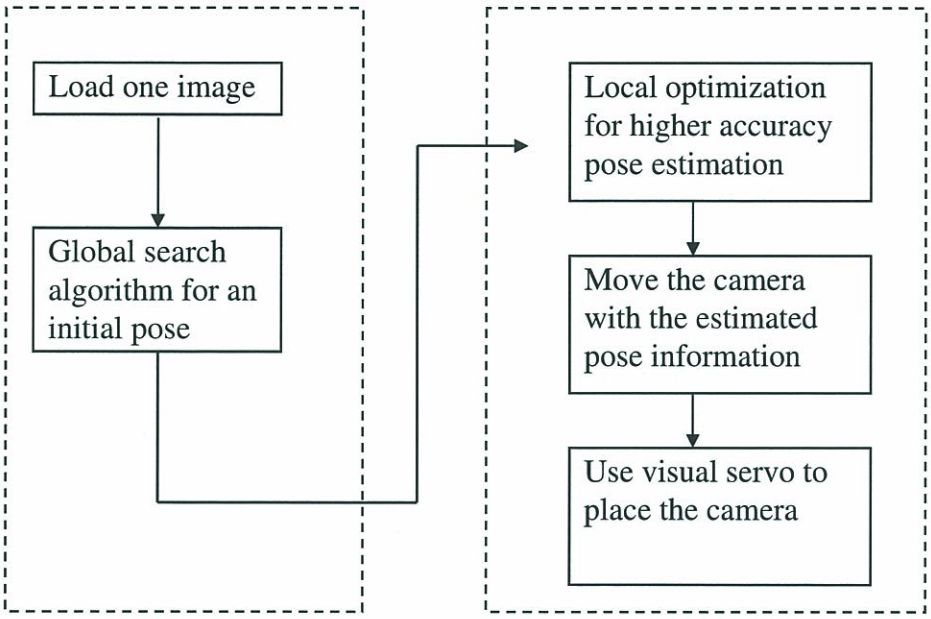
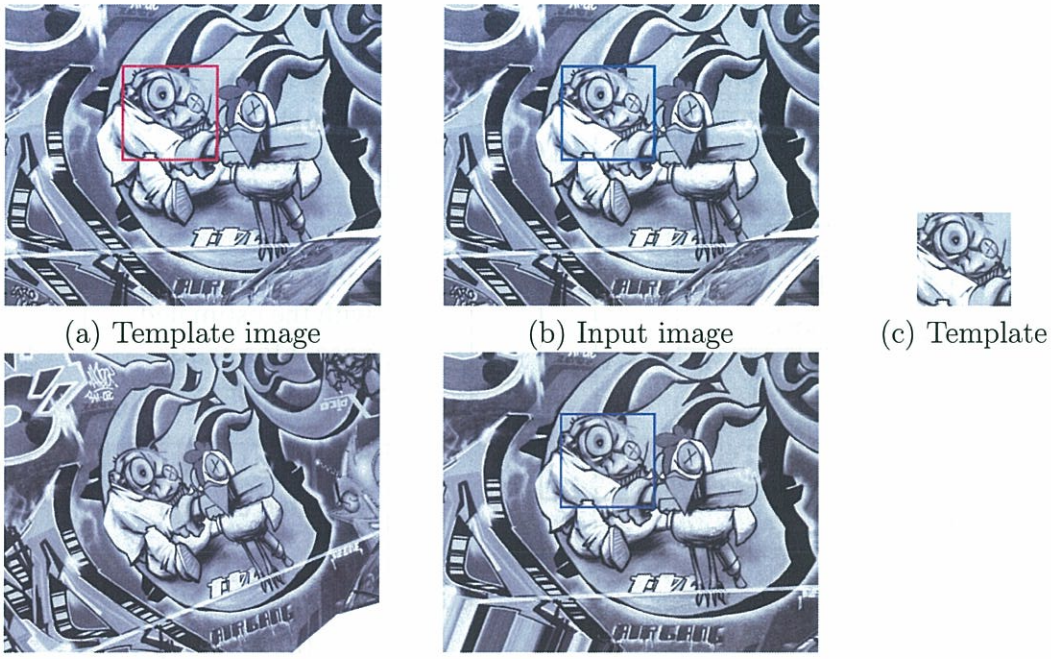


Figure 1.12: Process flowchart of a coarse-to-fine strategy adopted in placing a camera with respect to 3D objects. A global search algorithm offers coarse initial pose estimation in the coarse step. In the fine step, the pose estimation is refined by local optimization approaches. With this pose estimation the camera is moved to near its final pose. To reduce the error due to system calibration, a visual servo scheme is adopted finally to move the camera to its desired pose.

which is based on the Normalized Correlation Coefficient (NCC)

$$R(u,v) = \frac{\sum_{u',v'} \left(T(u',v')I(u+u',v+v')\right)}{\sqrt{\sum_{u',v'} T(u',v')^2 \sum_{u',v'} I(u+u',v+v')^2}}, \tag{1.11}$$

where u', v' are the pixel coordinates in the template patch T. u and v are the top left pixel coordinates of the current convolution region in the current image. By realizing this NCC convolution operation on the real image with the template patch, a maximum NCC value $R(u,v)$ can be found. If it is above a preset threshold (0.95 in our application), the inspection point is treated as good. Otherwise, the object will be treated as a faulty part.



(a) Template image (b) Input image (c) Template

(d) Input image (e) Warped image

Figure 1.13: Homography-based visual inspection with template matching. (a): The template image from which a template patch (c) is chosen (shown in the red rectangular). (b): One input image without rotation or scaling. The blue rectangular shows the matched region which has maximum similarity with the template patch. The similarity is used to determine the inspection region is Good or NG. (d) Another input image after rotation and scaling. This image can not be directly used for template matching. (e): The warped image by the homography-based inspection method by Inoue [126]. The template matching is carried out between the warped image and the template patch. The blue rectangular shows the matched region for further Good/NG inspection.

One limit of this method is that it performs poorly when the image is under rotation or scaling. To solve this problem, T. Inoue and K. Hashimoto have proposed a homography-based 2D inspection [126]. First the homography $\boldsymbol{G}$ between the input image and the template image is estimated and used to warp the input image to a warped image. Then the template matching is carried out between the warped image and the template patch. By adopt the homography-based strategy, the influence of rotation and scaling has been effectively removed and a more reliable matching result

can be obtained. In our system we adopt this homography-based inspection strategy. When the inspection camera is finally moved to its desired pose, the homography is estimated and used in the inspection implementations.

1.4 Contributions in Each Chapter

The main contributions of our study are as follows.

In chapter 2 we concentrate on using GPU acceleration to solve the slow processing speed problem of the ESM tracking algorithm, which causes it to fail in tracking fast moving objects or large tracking regions. We have analyzed the ESM algorithm and found the most time-costing part in the algorithm to optimize. In our GPU implementation we have adopted several GPU optimization techniques such as code parallelization, memory optimization, memory coalescing and strided access. The evaluation experiments show that, compared with the CPU-ESM implementation, our GPU implementation has accelerated it by 20 to 30 times and can realize the object tracking at high speed. The extension of the GPU optimization to the deformable image registration problem also shows that our GPU acceleration implementation is very useful for fast visual tracking applications.

In chapter 3 we have proposed a combination strategy for homography estimation so that it can be used in the homography-based visual servo to place a camera with 2D planar objects. In this combination strategy, the ESM tracking algorithm in chapter 2 provides fast homography estimation, meanwhile the initial homography, which is the key issue for the ESM tracking algorithm as a local optimization method, is provided by the SIFT matching algorithm. With this strategy, when the ESM tracking algorithm fails due to large initial error or occlusions, the homography from

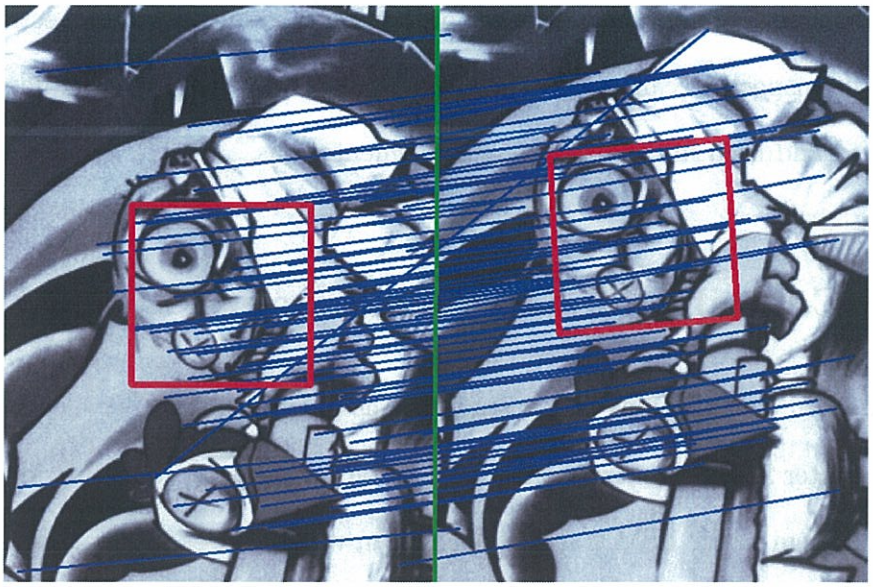
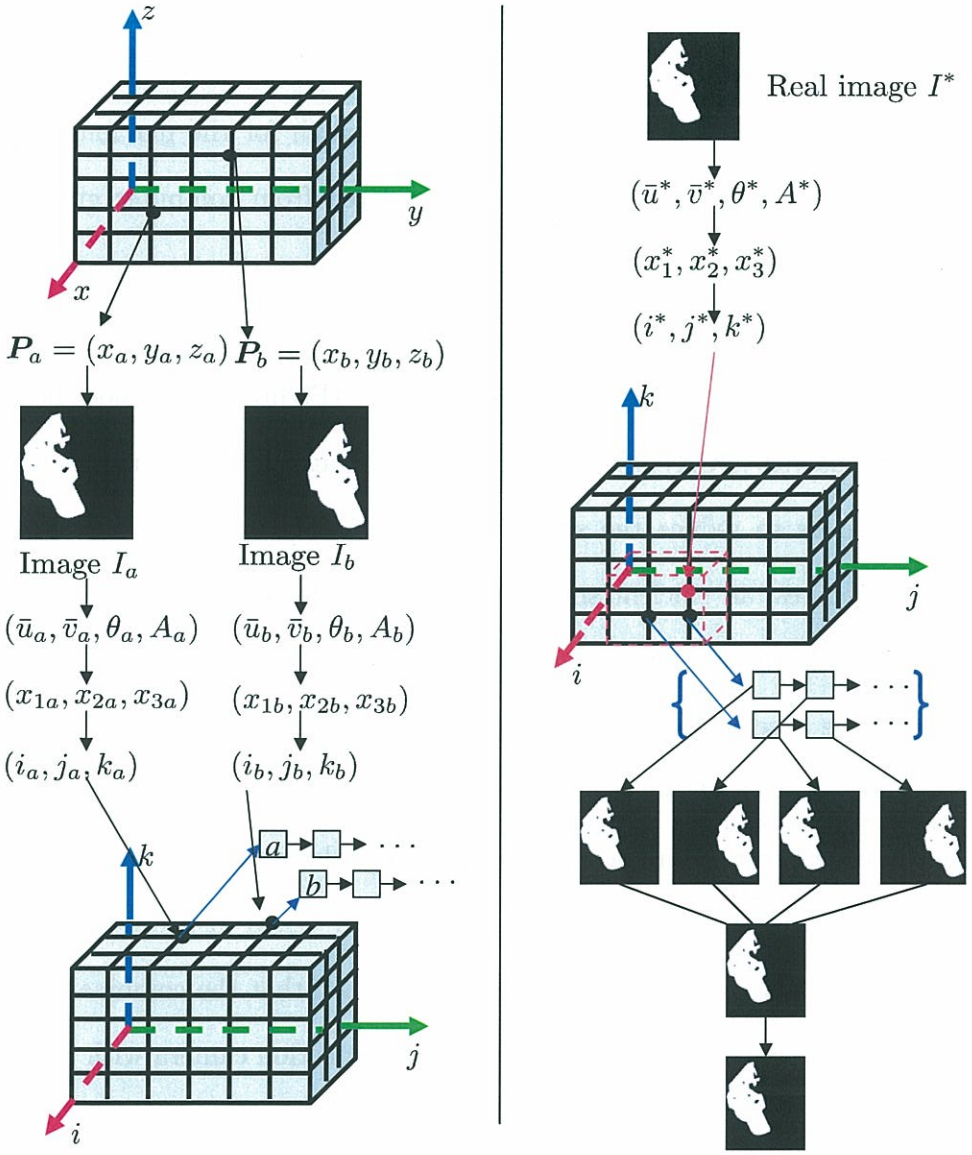


Figure 1.14: SIFT matching between two images. Each blue line represents one pair of matched SIFT features between these two images. From these corresponding points, a homography can be estimated and a tracking region can be distinguished as the red rectangular region in the two images. The input image is taken from the test dataset "Graffiti" by the *Robot Research Group* in University of Oxford [127].

the SIFT matching algorithm is loaded to restart the ESM tracking. By this means, the initial pose range has been enlarged and the system can continue placing the camera after occlusion happens.

We have adopted the multi-thread programming technique to run both GPU-ESM and GPU-SIFT algorithms on separate GPUs simultaneously to ensure high efficiency. We have proposed a switch strategy between both methods based on the ZNCC value when occlusion happens. Evaluation experiments show that, compared with the ESM or SIFT method, our combination strategy has improved the system performances on processing speed, convergence range and the robustness to occlusions. Therefore, the system can find homography estimation from large initial pose ranges at a high speed and robust to occlusions, which shows the combination strategy is very useful for the homography-based visual tracking and visual servo applications.

In chapter 4 we propose a coarse-to-fine strategy to place a camera with respect to 3D objects. We adopt an efficient view-based pose estimation method as the coarse step. The initial pose is estimated by a global search for the maximum similarity between the real image and those images rendered from computer graphics (CG) image, and the local optima as in other methods have been effectively avoided. Firstly, a series of image features are calculated from the image moments of all the CG images rendered from all possible poses in the search space (Figure 1.15). Then the principal component analysis (PCA) is carried out on these image features to reduce the feature dimension so as to reduce the size of the lookup table. With the reduced PCA projected features, an efficient small size lookup table is created. By searching the maximum similarity in a much smaller space (cell) in the lookup table, a remarkable speedup of the pose estimation has been obtained. The evaluation experiments show that, compared with those feature based methods which require a good guess of the initial pose, our combination strategy can find the pose estimation in large pose range without pose ambiguity. Real applications of visual inspection and pick-and-place operation with our pose estimation have shown that the combination strategy is an important contribution for positioning an inspection camera with 3D objects.



(a) Offline Learning

(b) Online Search

Figure 1.15: An example of global search with a lookup table for estimation of 3 DOF x, y, z of the pose (suppose orientation keeps the same). In the offline liearning process, two CG images I_a and I_b are rendered from two pose $\boldsymbol{P}_a$ and $\boldsymbol{P}_b$. Then features $(\bar{u}, \bar{v}, \theta, A)$ are calculated and projected to (x_1, x_2, x_3) with PCA projection. A 3-element integer index (i, j, k) is calculated from the features (x_1, x_2, x_3) and the pose index a and b are filled in the 3D lookup table. In the online search process, the features are extracted from the real image I^* as the same way. Then a cell around the index (i^*, j^*, k^*) are chosen and a group of candidate poses are extracted. Then for each candidate pose, a CG image is rendered and compared with the real image. The pose to produce the CG image with maximum similarity as the real image is treated as the poes estimation.

1.5 Dissertation Outline

This dissertation is organized as follows.

Chapter 1 describes our study objectives and motivations. A survey of the related approaches for camera positioning is presented in detail.

Chapter 2 presents our work on the GPU acceleration of the ESM tracking algorithm. The algorithm and its GPU implementation are introduced in detail. We also extend the GPU acceleration to theTPS deformable image registration. Evaluation experiments and optimization techniques in CUDA applications have been presented.

Chapter 3 describes our combination strategy of the GPU-ESM and GPU-SIFT algorithms to place a camera with 2D planar objects. The combination strategy in homography-based visual servo is introduced in detail. Experiments are presented to evaluate the efficiency. The multi-thread programming model and switch strategy between both algorithms are described in this chapter.

Chapter 4 describes our coarse-to-fine strategy for placing a camera relative to 3D objects. The combination and system implementation are introduced. For the coarse step, the global search method with a PCA processed lookup table to reduce the search time and memory space is described in detail. For the fine step, several local optimization approaches such as least square techniques and edge-base pose estimation are introduced in detail. Experiments with various 3D objects are presented to show the efficiency of our strategy.

Chapter 5 concludes this dissertation and a discussion of future work is proposed.

Finally, the mathematics tools and robot kinematics commonly used in this dissertation are described in the appendix parts.

Chapter 2

GPU Accelerated Efficient Second order Minimization Tracking

This chapter concentrates on our GPU accelerated Efficient Second-order Minimization (GPU-ESM) algorithm to ensure a fast homography solution, which is essential for the homography-based applications, such as visual tracking, visual servo to place a camera with respect to 2D planar objects. The implementation of our GPU applications is described in this chapter. The efficiency of the GPU acceleration is evaluated in the visual tracking experiments of planar objects. The optimization techniques of our GPU acceleration are discussed in detail.

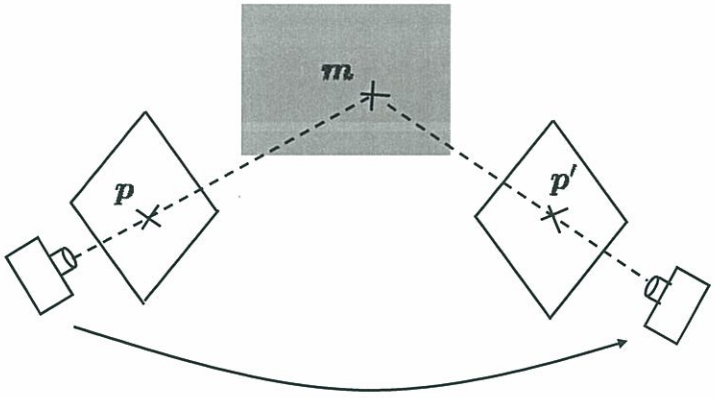


Figure 2.1: The homography between two images of a planar object. The image points $\boldsymbol{p}$ and $\boldsymbol{p}'$ are of the same 3D point $\boldsymbol{m}$. There is the homography relationship between the two image points.

2.1 Introduction

2.1.1 Homography-based Applications

Homography is an important issue in the images of a planar object. An illustration of homography is shown in Figure 2.1. Suppose the camera is placed at two different poses and get two images of the same planar objects, there is a homography relationship between the corresponding pairs of images points in the two images, here the corresponding means the two images points are of the same 3D point in the image. For the image points $\boldsymbol{p} = [u, v]^\top$ and $\boldsymbol{p}' = [u', v']^\top$ of the same 3D point $\boldsymbol{m}$, the relationship can be expressed in the homogeneous coordinates by

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{G}_{3\times 3} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}, \tag{2.1}$$

where s is a scale and $\boldsymbol{G}$ is the homography matrix.

(a) Reference image (b) Input image

Figure 2.2: Visual tracking in two images of a planar object. The region inside the red rectangular in a reference image (a) is chosen as the target region for tracking. Visual tracking means finding the same region in other input images. For instance, the area inside the red rectangular in (b) shows the image points of the target region in this input image. The images are taken from the test dataset "Graffiti" by the *Robot Research Group* in University of Oxford [127].

As all the corresponding pairs of image points have the same homography $\boldsymbol{G}$, this property has invoked many applications in computer vision applications, such as visual servo and visual tracking. A survey of homography-based visual servo is given in section 1.2.2. In this chapter we focus on the application of homography-based visual tracking of a planar object. Visual tracking means using computer vision techniques to find the image points of a certain interested region of an object so that the further image processing can be carried out with this image points (Figure 2.2).

Generally speaking, the homography solution is a typical minimization problem of sum of squared differences between a region in a reference image and a warped region in a current image [84]. Many nonlinear optimization approaches have been proposed to deal with it with different kinds of approximations, such as Standard Newton method, Gauss-Newton approximation [128]. Among these solutions, the Efficient Second-order Minimization (ESM) algorithm is an elegant idea which obtains

the same convergence speed as standard Newton method while not computing the computationally costly Hessian matrix [129]. Based on the ESM algorithm, Malis has realized the homography-based visual tracking and visual servo [47].

2.1.2 Fast Implementation of Homography Estimation

Though the ESM algorithm has shown great application potential, when applied in a realtime visual servo system, the algorithm efficiency still needs improving. From our practical experience, with a large tracking region size (for example 360×360 pixels), the ESM computation still takes too much time and induces a relative low processing speed. The processing speed is the key issue to improve the tracking efficiency. As the camera is mounted on a moving robot, this low speed will cause a larger image difference in the two consecutive images. Therefore only small overlapping region exists in this pair of images. The ESM algorithm will fail to find the right homography because it can not obtain enough information from this relative small overlapping region. Consequently, it is necessary to accelerate the ESM algorithm to ensure better system performances. Ito and Deguch have proposed a GPU accelerated ESM implementation [130]. The processing speed has been accelerated by using the GPU's parallel processing capability.

2.1.3 Goals and Contributions

In this chapter our goal is to realize fast homography estimation so that we can realize fast visual tracking applications. To deal with those problems mentioned above, we have proposed a novel idea of using GPUs to improve the system performance. Our contributions are mainly as follows.

We have proposed a GPU accelerated ESM algorithm to address the need for faster homography solutions in the homography-based visual servo. In the GPU implementation of the ESM algorithm, we have realized the whole processing steps on the GPU to reduce the intermediate data transportation between the CPU and the GPU, so that the running time is also reduced. With carefully parallelization and optimization for the ESM algorithm in CUDA, our GPU version ESM algorithm can work at about $20 - 30$ times faster than its CPU version. This speedup allows for a higher speed camera, with which there will be smaller difference for the same motion between the two consecutive frames. Therefore, the system performance has been greatly improved by this speedup.

Besides the ESM algorithm, we have extended the optimization techniques to the Thin-Plate Spline (TPS) deformable image registration. The evaluation experiments have shown the efficiency of our optimization techniques.

2.1.4 Outline of the Rest Parts

The rest of this chapter is organized as follows. Section 2.2 describes the structure of the ESM algorithm with reviews of the related works. Section 2.3 introduces the translation details of the GPU-ESM algorithms so as to fully utilize the parallel architecture of GPUs. Section 2.5 describes the experimental results to evaluate our proposed algorithm. Section 2.4 describes the key optimization techniques in our GPU applications to obtain the speedup. Section 2.6 concludes this chapter.

2.2 Structure of the ESM Algorithm

The ESM algorithm was proposed by Malis in 2004 [131]. By performing a second order approximation of a SSD problem with only first order derivatives, the ESM algorithm can get a high convergence rate and avoid local minima close to the right global one. Because of these merits, the ESM algorithm has been reported in many different kinds of applications, such as visual tracking of planar objects [47] and deformable objects [132], visual servo [48] etc.

Suppose a planar object is projected in a reference image I^* with a "Template" region of q pixels (Figure 2.3). Tracking this template in the current image I consists in finding the homography transformation $\boldsymbol{G}$ that transforms each pixel P_i^* of the "Template" into its corresponding pixel in the current image I, i.e. finding the homography $\boldsymbol{G}$ such that $\forall i \in \{1, 2, ...q\}$,

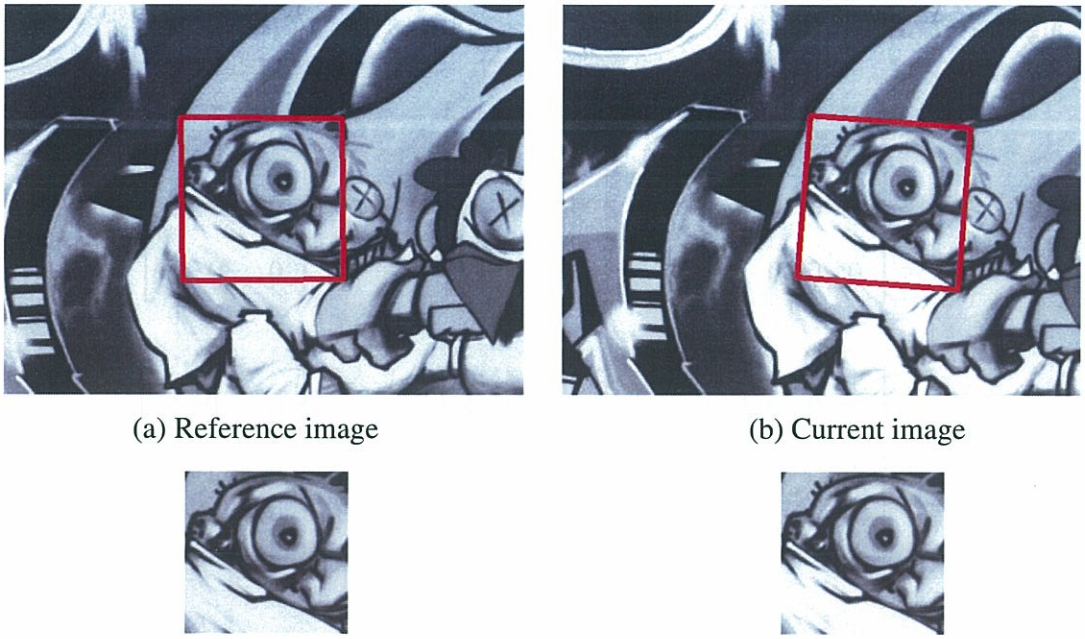
$$I(w(\boldsymbol{G})(P_i^*)) = I^*(P_i^*), \tag{2.2}$$

where $P_i^* = [u^* v^* 1]^\top$ is the homogeneous coordinate of a pixel. The homography $\boldsymbol{G}$ is defined in the Special Linear group $\mathbb{SL}(3)$. The $\boldsymbol{G}$ defines a projective transformation between the images. Let w be a group action defined from $\mathbb{SL}(3)$ on $\mathbb{P}^2$

$$w : \mathbb{SL}(3) \times \mathbb{P}^2 = \mathbb{P}^2. \tag{2.3}$$

Therefore, for all $\boldsymbol{G} \in \mathbb{SL}(3)$, $w(\boldsymbol{G})$ is a $\mathbb{P}^2$ automorphism:

$$\begin{aligned} w(\boldsymbol{G}) : \ \mathbb{P}^2 &\rightarrow \mathbb{P}^2, \\ P^* &\rightarrow P = w(\boldsymbol{G})(P^*), \end{aligned} \tag{2.4}$$



(a) Reference image (b) Current image

(c) Template region (d) Warped region

Figure 2.3: The homography between two images of a planar object. (a): Reference image, (b): Current image, (c): Template region I^* which is distinguished by a red rectangular, (d):Warped region from the current image with homography $\boldsymbol{G}$. The tracked region is also drawn in a red rectangular in (b). The input images are taken from the test dataset "Graffiti" by the *Robot Research Group* in University of Oxford.

such that

$$P = w(\boldsymbol{G})(P^*) = \begin{bmatrix} \frac{g_{11}u^* + g_{12}v^* + g_{13}}{g_{31}u^* + g_{32}v^* + g_{33}} \\ \frac{g_{21}u^* + g_{22}v^* + g_{23}}{g_{31}u^* + g_{32}v^* + g_{33}} \\ 1 \end{bmatrix}. \tag{2.5}$$

Suppose that we have an approximation $\widehat{\boldsymbol{G}}$ of $\boldsymbol{G}$, the problem consists in finding an incremental transformation $\Delta\boldsymbol{G}$ such that the difference between a region of current image I (transformed with the composition $w(\widehat{\boldsymbol{G}}) \circ w(\Delta\boldsymbol{G})$) and the corresponding region in the image I^* is null.

The homography $\boldsymbol{G}$ belongs to the $\mathbb{SL}(3)$ group which is a Lie group. The Lie algebra associated to this group is $SL(3)$. Let $\{\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_8\}$ be a basis of the Lie

algebra $SL(3)$:

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{A}_4 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

$$\boldsymbol{A}_7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_8 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Then a matrix $\boldsymbol{A}(\boldsymbol{x})$ in the $\mathbb{SL}(3)$ group can be expressed as follows

$$\boldsymbol{A}(\boldsymbol{x}) = \sum_{i=1}^{8} x_i \boldsymbol{A}_i. \tag{2.7}$$

A homography $\boldsymbol{G}(\boldsymbol{x}) \in \mathbb{SL}(3)$ in the neighborhood of $\boldsymbol{I}$ can be parameterized as

$$\boldsymbol{G}(\boldsymbol{x}) = \exp(\boldsymbol{A}(\boldsymbol{x})) = \sum_{i=0}^{\infty} \frac{1}{i!} (\boldsymbol{A}(\boldsymbol{x}))^i. \tag{2.8}$$

As the incremental transformation $\Delta\boldsymbol{G}$ also belongs to $\mathbb{SL}(3)$, it can be expressed as $\Delta\boldsymbol{G}(\boldsymbol{x})$, where $\boldsymbol{x}$ is a 8×1 vector. Therefore tracking consists in finding a vector

$\boldsymbol{x}$ such that $\forall i \in \{1, 2, ...q\}$, the image difference

$$y_i(\boldsymbol{x}) = I((w(\widehat{\boldsymbol{G}}) \circ w(\Delta \boldsymbol{G}(\boldsymbol{x})))(P_i^*)) - I^*(P_i^*) = 0. \tag{2.9}$$

Using a vector to express the $q \times 1$ pixel differences $\boldsymbol{y}(\boldsymbol{x}) = [y_1(\boldsymbol{x}), y_2(\boldsymbol{x}), ...y_q(\boldsymbol{x})]^\top$, then the problem consists in finding a $\boldsymbol{x} = \boldsymbol{x}_0$ verifying

$$\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{0}. \tag{2.10}$$

Linearizing $\boldsymbol{y}(\boldsymbol{x})$ around $\boldsymbol{x} = \boldsymbol{0}$ with a second-order Taylor series approximation

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{0}) + \boldsymbol{J}(\boldsymbol{0})\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}(\boldsymbol{0})\boldsymbol{x} + \boldsymbol{O}(\|\boldsymbol{x}\|^3), \tag{2.11}$$

where $\boldsymbol{J}(\boldsymbol{0})$ is the Jacobian matrix. In the ESM algorithm, the Hessian matrices of $\boldsymbol{y}(\boldsymbol{x})$ are replaced by a first-order Taylor Series approximation of $\boldsymbol{J}(\boldsymbol{x})$ about $\boldsymbol{x} = \boldsymbol{0}$

$$\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{0}) + \boldsymbol{x}^\top \boldsymbol{H}(\boldsymbol{0}) + \boldsymbol{O}(\|\boldsymbol{x}\|^2). \tag{2.12}$$

Then (2.11) becomes

$$\boldsymbol{y}(\boldsymbol{x}) \approx \boldsymbol{y}(\boldsymbol{0}) + \frac{1}{2}(\boldsymbol{J}(\boldsymbol{0}) + \boldsymbol{J}(\boldsymbol{x}))\boldsymbol{x}. \tag{2.13}$$

For $\boldsymbol{x} = \boldsymbol{x}_0$, we have

$$\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{y}(\boldsymbol{0}) + \frac{1}{2}(\boldsymbol{J}(\boldsymbol{0}) + \boldsymbol{J}(\boldsymbol{x}_0))\boldsymbol{x}_0 = \boldsymbol{0}. \tag{2.14}$$

With some mathematical proofs by Malis, the sum of Jacobian matrix $\frac{1}{2}(\boldsymbol{J}(\mathbf{0}) + \boldsymbol{J}(\boldsymbol{x}_0))$ can be written as one matrix $\boldsymbol{J}_{esm}$. Therefore, for $\boldsymbol{x} = \boldsymbol{x}_0$, we have

$$\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{y}(\mathbf{0}) + \boldsymbol{J}_{esm}\boldsymbol{x}_0 = \mathbf{0}. \tag{2.15}$$

The solution $\boldsymbol{x}_0$ can be obtained by

$$\boldsymbol{x}_0 = \boldsymbol{J}_{esm}^{+}\boldsymbol{y}(\mathbf{0}). \tag{2.16}$$

The incremental transformation $\Delta\boldsymbol{G}(\boldsymbol{x}_0)$ can be calculated with the Lie Algebra operation by (2.7) and (2.8). Then a new homography solution $\boldsymbol{G}$ can be obtained by such update process

$$\boldsymbol{G} = \widehat{\boldsymbol{G}}\Delta\boldsymbol{G}(\boldsymbol{x}_0). \tag{2.17}$$

2.3 Implementation of the GPU-ESM Algorithm

2.3.1 Hardware and Software Configuration

The computation platform is a desktop PC with Intel Core i7-920 (2.67 GHz), 3GB RAM and a NVIDIA GTX295 graphic board (Figure 2.4). The GTX295 board integrates two GTX280 GPUs inside and has 896MB GPU RAM for each GTX280 GPU. The operating system is Windows XP (sp2). We implement our GPU-ESM algorithms with NVIDIA's CUDA (Compute Capability 1.3). In CUDA's programming model, the functions are expressed as kernels and the smallest execution unit on GPU is a thread. As one CUDA kernel just completes a certain task like an ordinary C function, usually multiple CUDA kernels are needed to realize different kinds of

(a) GPU (GTX-295)

CUDA Cores:	480 (240 per GPU)
Graphics Clock:	576 MHz
Processor Clock:	1242 MHz
Standard Memory	1792 MB GDDR3

(b) Specifications

Figure 2.4: In our system the NVDIA's GTX-295 graphic board is used. It is composed of two GTX-280 GPU cores.

functions in one algorithm.

2.3.2 Implementation of GPU-ESM Algorithm

The whole process flowchart of the ESM algorithm is shown in Figure 2.5. According to this process flow, we realize our GPU-ESM algorithm with seven CUDA kernels. To accelerate the ESM algorithm and avoid redundant computations, several steps in the algorithm only process the ROI part of the image (with "**ROI**" in the end of these steps in the figure).

1. **Image Warping**. This kernel completes the task of warping the current image I^* to the a warped image I_w with the current homography.
2. **Gradient Calculation**. This kernel calculates the image intensity gradient in the u and v directions of the warped image I_w and the reference image I^*. The computation for the reference image I^* is processed only once as it does not change in the whole process.
3. **Jesm Calculation**. This kernel calculates the $\boldsymbol{J}_{esm}$ matrix in the ESM algorithm from the previous image gradients computation. As the homography is

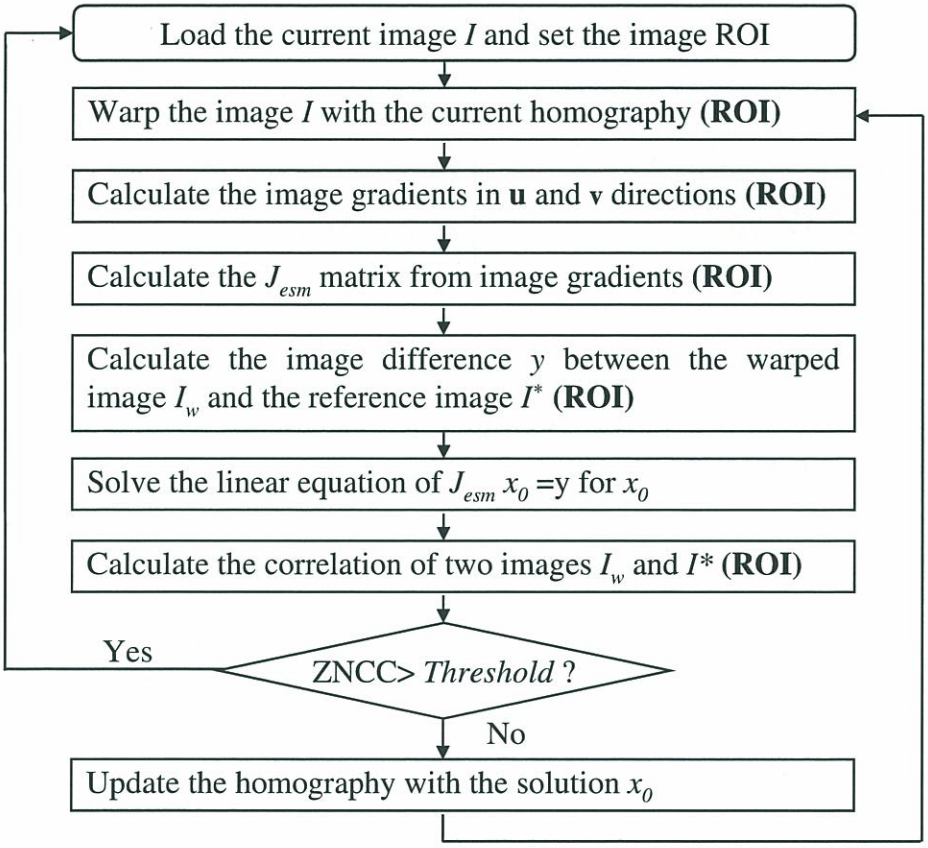


Figure 2.5: The process flowchart of the ESM algorithm. Those steps with **ROI** in the end only process the Region of Interest (ROI) of the whole image.

parameterized by 8×1 parameters $\boldsymbol{x}$, this kernel calculate a 1×8 vector for each pixel in the ROI. Therefore, the $\boldsymbol{J}_{esm}$ matrix is of $q \times 8$.

4. **Difference Calculation**. This kernel calculates the image difference $\boldsymbol{y}$ between the warped image I_w and the reference image I^*.

5. **Equation Solving**. This kernel finds the solution $\boldsymbol{x}_0$ of the equation

$$\boldsymbol{J}_{esm}\boldsymbol{x}_0 = \boldsymbol{y}, \tag{2.18}$$

where $\boldsymbol{J}_{esm}$ is of $q \times 8$ and $\boldsymbol{x}_0$ is of 8×1, therefore this equation is overdetermined.

To solve this equation, we multiply the transpose of $\boldsymbol{J}_{esm}$ on both sides

$$\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}\boldsymbol{x}_0 = \boldsymbol{J}_{esm}^{\top}\boldsymbol{y}, \tag{2.19}$$

and adopt the Cholesky decomposition method to solve the $\boldsymbol{x}_0$ in (2.19).

6. **Correlation Calculation**. This kernel calculates the correlation of warped image I_w and template region I^* (q pixels). Here we use the Zero mean Normalized Cross Correlation (ZNCC) as the correlation standard:

$$d = \frac{\sum_{k=1}^{q}\left(I_w(k) - \overline{I_w}\right)\left(I^*(k) - \overline{I^*}\right)}{\sqrt{\sum_{k=1}^{q}\left(I_w(k) - \overline{I_w}\right)^2 \sum_{k=1}^{q}\left(I^*(k) - \overline{I^*}\right)^2}} \tag{2.20}$$

where $\overline{I_w}$ and $\overline{I^*}$ are the mean intensity values of the warped region I_w and the template region I^*, respectively. As a quantitative evaluation criterion for the tracking accuracy, the correlation plays two important roles in our application. On one hand, if the correlation is larger than a preset upper threshold, the current solution will be treaded as the final solution and the iterative ESM loops will stop and continue to process the next input image. As an iterative minimization method, such threshold is necessary to stop the loops. On the other hand, if the correlation is smaller than a preset lower threshold, it will be treated as ESM tracking failure. In this case, a robust homography will be offered by SIFT. This part will be introduced later in chapter 3. Due to the illumination changes and different kinds of objects, these two threshold values are chosen manually from practical experiments.

7. **Pose Update**. This kernel updates the homography with the solution $\boldsymbol{x}_0$ by (2.7), (2.8) and (2.17). We adopt such approximation to calculate the matrix

exponential of $\exp(\boldsymbol{A}(\boldsymbol{x}_0))$

$$\begin{aligned} \boldsymbol{G}(\boldsymbol{x}_0) &= \exp(\boldsymbol{A}(\boldsymbol{x}_0)) = \sum_{i=0}^{\infty} \frac{1}{i!}(\boldsymbol{A}(\boldsymbol{x}_0))^i \\ &\approx \boldsymbol{I} + \boldsymbol{A}(\boldsymbol{x}_0) + \frac{1}{2}\boldsymbol{A}(\boldsymbol{x}_0)^2 + \frac{1}{6}\boldsymbol{A}(\boldsymbol{x}_0)^3. \end{aligned} \tag{2.21}$$

Due to the small eigenvalues of $\boldsymbol{A}(\boldsymbol{x}_0)$ (near 0), above approximation can work well without loosing accuracy.

2.4 Optimization Techniques

In this section, we share our optimization experience during our GPU application implementation. To make the GPU code highly proficient, carefully optimization must be exploited and several important factors must be considered. We will describe our optimization strategies with code parallelization, memory hierarchy and memory access in detail.

2.4.1 Code Parallelization

Algorithms must be carefully parallelized so that they can fully use the parallelism of a GPU, otherwise no obvious speedup can be obtained. Amdahl's law specifies the maximum speedup (s) that can be expected by parallelizing portions of a sequential program as

$$s = \frac{1}{(1-P) + \frac{P}{N}} \tag{2.22}$$

where P is the fraction of the total serial execution time taken by the portion of code that can be parallelized, N is the number of processors on which the parallel portion code runs. When P is small, i.e. the code is not effectively parallelized, no matter

how many cores you use (N), there will be little improvement. So maximizing the amount of code that can be parallelized is the most important factor. For simplicity, we only take the matrix multiplication in our application to show the important role of the code parallelization.

With *CUDA Profiler* we find that the kernel "Equation Solving" takes most of the running time. In this kernel, we need to solve the equation (2.18)

$$\boldsymbol{J}_{esm}\boldsymbol{x}_0 = \boldsymbol{y}.$$

In the beginning, we directly use the *CULA*, a third-party library for Linear Algebra Package interface for CUDA to solve the problem. In CULA the function *culaDeviceSgels* can directly get the solution with QR or LQ factorization of $\boldsymbol{J}_{esm}$. However it is not fast enough for our application. For 300×300 pixels region, it costs 50 ms to solve it. As each image needs several iteration loops, the fps is very low (< 4 fps). Hence we turn to calculate the equation ((2.19)) instead:

$$\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}\boldsymbol{x}_0 = \boldsymbol{J}_{esm}^{\top}\boldsymbol{y}_0,$$

where $\boldsymbol{J}_{esm}$ is of $q \times 8$, $\boldsymbol{y}_0$ and $\boldsymbol{x}_0$ are of 8×1, q is the number of pixels in the template region. Therefore, GPU implementation of matrix multiplication ($\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}$) and ($\boldsymbol{J}_{esm}^{\top}\boldsymbol{y}_0$) is necessary. At first we tried the CUBLAS library. It is simple to use but the result is not satisfactory (Figure 2.6). Therefore specific CUDA kernels are needed to obtain a better performance. We present two block matrix multiplication methods and compare the running time in Figure 2.6.

In method 1, each CUDA block is composed of 512 threads and each block calcu-

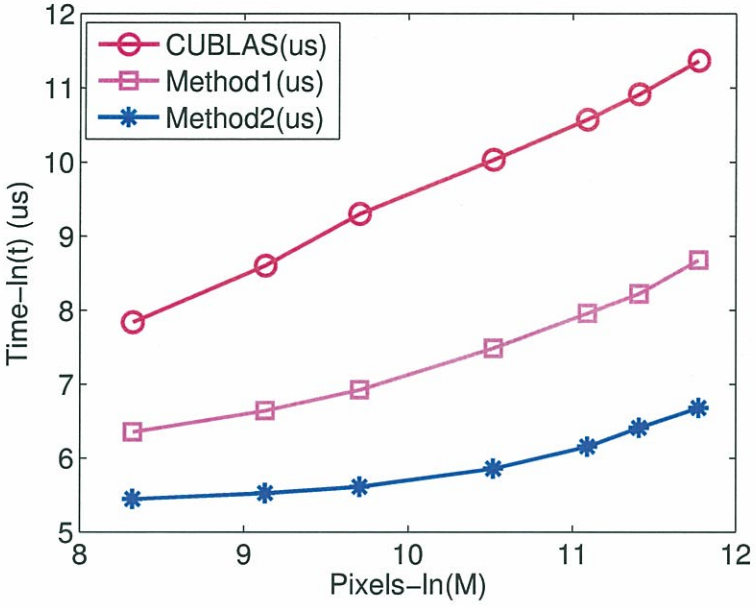


Figure 2.6: Running time comparison of three methods with respect to various processing region size. The natural logarithmic scales are used on both the horizontal and vertical axes.

lates a block matrix multiplication (8×64 by 64×8) to get an 8×8 intermediate result (Figure 2.7). After all the blocks obtaining their results, a second summary kernel will calculate the final result by summarizing all the intermediate results.

In method 2, each block contains 256 threads and calculates 1 element of the result (Figure 2.8). As the total number of the elements in $\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}$ is 64, the number of blocks is set to 64.

Results in Figure 2.6 show that method 2 is even faster than method 1 though both run on the same GPU. To investigate the difference in detail, we take $q = 300 \times 300 = 90000$ for an example. The block number in method 1 is $90000/64 \approx 1407$ and two steps are needed to get the final result. Meanwhile method 2 only needs 64 blocks and can get the result in one step. Limited by CUDA's compute capability, a GTX280 GPU can run at most 120 blocks simultaneously when one block contains 256 threads.

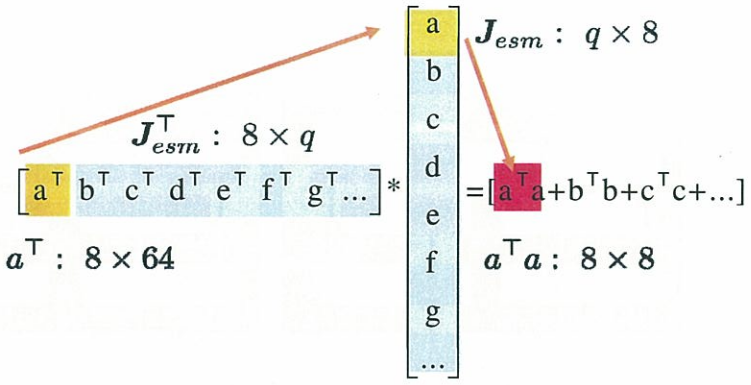


Figure 2.7: Method 1: Each block calculates a block matrix multiplication (8×64 by 64×8) to get an 8×8 intermediate result.

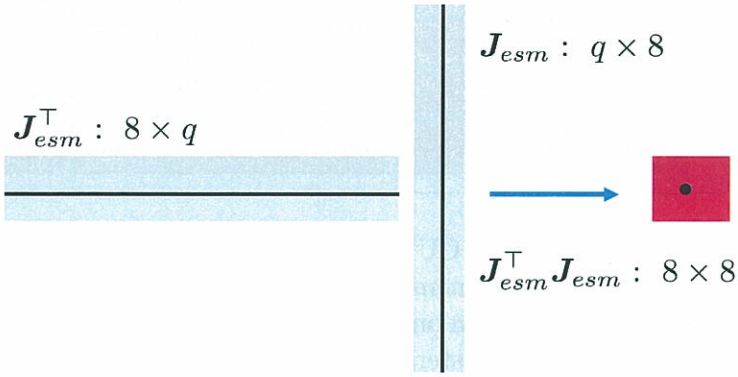


Figure 2.8: Method 2: Each block contains 256 threads and calculates 1 element of the result by multiplying one row in $J_{esm}^{\top}$ and one column in J_{esm}.

So in method 2, all these 64 blocks can run parallel on a GPU. While in the method 1, it needs block scheduling because of so large a block number (1407) and it has to save all the 1407 intermediate results in global memory and load them again for next summary operation. As we will mention later, this kind of global memory fetching takes too much time. The method 1 essentially does not fully parallelize the code as some blocks have to wait for the accomplishment of other blocks during the block scheduling. Therefore, with fully code parallelization in method 2, significant speedup has been obtained.

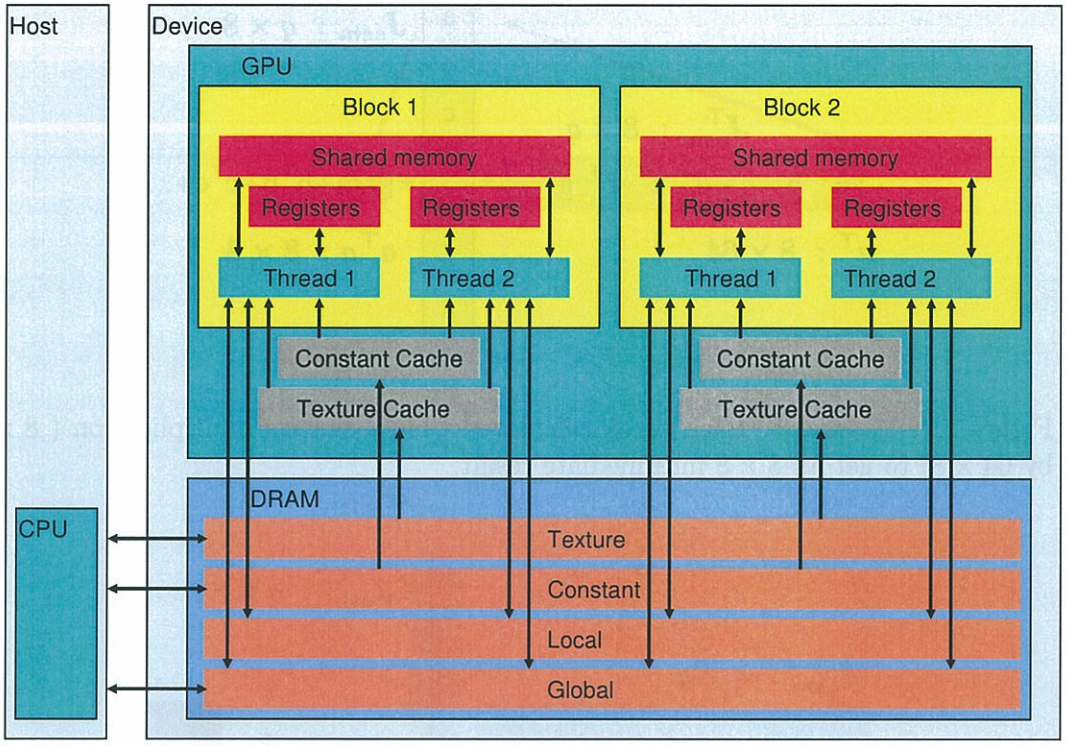


Figure 2.9: An illustration of the CUDA memory hierarchy. Global memory, constant memory, texture memory, local memory lie in the off-chip memory (DRAM). Shared memory lies in each block and can only be visited by those threads in the block. Each thread is also assigned with registers and local memory (in the DRAM). The word "Device" means the whole GPU board (GTX 295 in our system) while "Host" in the figure means the CPU and main memory system.

2.4.2 Memory Optimization

Memory optimization mainly consists in using different kinds of GPU memory to accelerate the application. Figure 2.9 shows the CUDA memory hierarchy, which consists of texture memory, constant memory, local memory, global memory, shared memory, and registers. Accessing the off-chip memory will takes much more time than accessing the on-chip memory. For example, it takes $400 - 600$ clock cycles to access the global memory or local memory, while the on-chip local memory or register accessing can be 100X faster than it.

In our applications, we intensively utilize the fast shared memory instead of the long-latency global memory to reduce the running time.

In the kernel "Jesm Calculation", the computation of $\boldsymbol{J}_{esm}$ matrix needs several intermediate results based on the image gradient. So we first load the image gradient data into the shared memory of each block and then continue other computation on them. By using this "cache" like strategy, this kernel's processing time is reduced from 300 us to 50 us.

Constant memory has been cached by the GPU processors, therefore this kind of memory is used in our applications for those frequently read constants. We also use texture memory in the kernel "Image Warping" because CUDA provides such bilinear filter function. When fetching the texture memory, the returned value is computed automatically based on the input coordinates. Though the running time is similar to our own kernel but with this hardware function we can skip its programming. We only need to set the filter mode to bilinear.

2.4.3 Memory Coalescing and Strided Access

By memory coalescing in CUDA, a half warp of 16 GPU threads can complete 16 global data fetching in as few as 1 or 2 transactions if certain memory access requirements are met. Figure 2.10 shows two examples of this memory access. In the coalesced access case (a), for the 16 threads in a half warp, if the $k\ th$ thread is accessing the $k\ th$ data, fetching 16 data can be finished in one transaction. Meanwhile, in the un-coalesced case (b), the data is fetched by two transactions. Hence, in our application, we have tried to avoid the un-coalesced cased to reduce the whole transaction time.

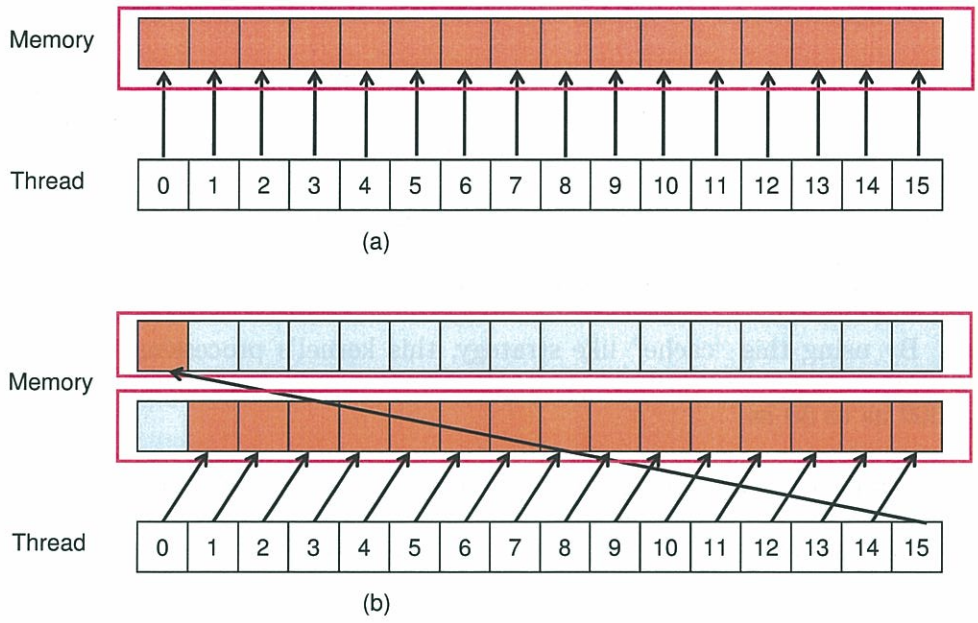


Figure 2.10: Memory coalescing. (a) Coalesced access takes only one transaction, where all 16 threads access the corresponding 16 data in a segment. (b) Un-coalesced access takes two transaction to fetch all the 16 data on GTX 280. For other GPUs (e.g., FX 5600), it takes 16 transactions. One red rectangle means one memory access. The data to be fetched is shown in orange color.

Besides the memory coalescing, we also intensively use the strided access technique. Figure 2.10 (a) shows the strided access with stride = 1, i.e., successive threads in a GPU block access the successive memory locations. Using strided memory access can For example, to calculate the mean $\overline{I}$ in ZNCC correlation, we need to calculate the sum of I. We use 1 block of 512 threads (the thread index "threadID" is from 0 to 511) to accumulate all the 90000 pixels. As $90000 = 176 \times 511 + 64$, the number of data processed by each thread is 176 (except the last thread with only 64 data). One idea is using such "for-loop" in each GPU thread. That is, for the thread with ID "threadID":

```
for (k=threadID*176;k<(threadID+1)*176;k++) sum+=I[k];
```

To use strided access, we change the code to follows:

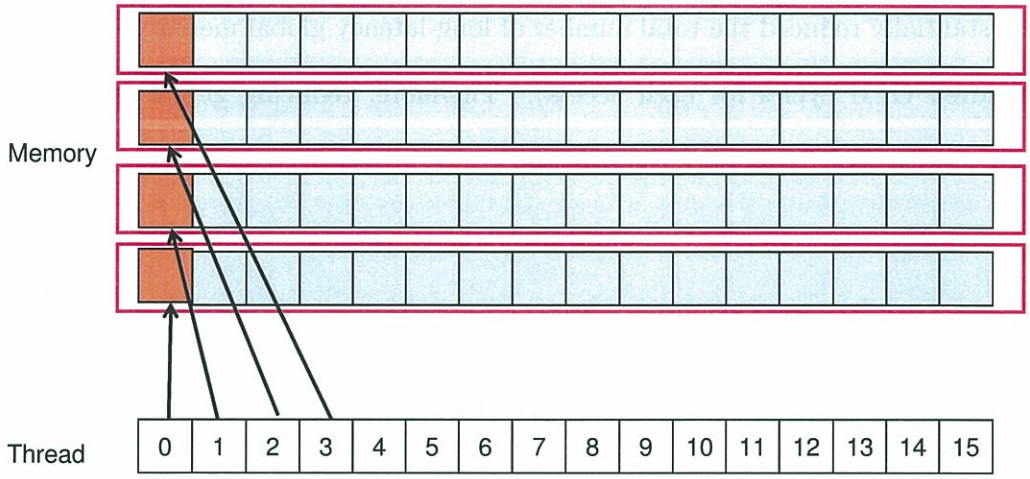


Figure 2.11: An illustration of strided memory access with stride = 176. The data to be fetched (in orange color) lie in different alignment in the memory space, therefore, it is necessary to invoke 16 memory accesses to fetch the 16 data for the 16 threads. In contrast, in Figure 2.10(a) with stride = 1, the 16 data can be fetched by only one memory access.

```
for (k=threadID;k<90000;k+=512) sum+=I[k];
```

Though both "for-loop"s have same performances on a CPU thread, speedup really happens on the GPU threads because of the GPU's particular memory fetching technique. GPU memory is accessed in a specific block mode, i.e., during one GPU memory access, the successive data from a block of continuously addressing memory locations will be loaded simultaneously. For example, it can load $I[0] \sim I[15]$ simultaneously by one GPU fetch (Figure 2.10 (a)). In the latter method, the loaded 16 data can be processed in parallel by 16 GPU threads. Meanwhile, in the former method, only 1 data of these fetched 16 data is used in 1 thread while all the other 15 data is deserted (Figure 2.11). Each of the other 15 threads must invoke another 15 GPU memory access to fetch these data. Therefore, for the same data fetching, the former method needs about 15 times more memory access than the latter method does.

By using strided memory access strategy shown in the latter method, we have

substantially reduced the total number of long-latency global memory access (several hundred GPU cycles for each access). Therefore, reducing global memory access provides us with great performance gain.

2.5 Experiments for Efficiency Evaluation

Experiments have been carried out to evaluate the system performances. The first experiment gives the running-time proportions of all parts of both ESM algorithms. The second one is carried out to show the efficiency of our GPU-ESM algorithm. The third experiment is based on the significant speedup from the GPUs. We have implemented the ESM tracking for multiple planes. In the fourth experiment we extend our GPU optimization techniques to the TPS deformable image registration problem. The experimental images in the first three experiments are captured from a 200 fps camera (*Grasshopper GRAS-03K2M/C*). Size is 640×480.

2.5.1 Experiment I: Running Time Proportions

We compare our GPU-ESM algorithm with the CPU-ESM algorithm on the same desktop. The running-time proportion of each CUDA kernel is listed in Table 2.1. The template area is set to $q = 300 \times 300$ pixels. The GPU running time is provided by the software *CUDA Visual Profiler* from NVIDIA. The CPU running time is counted by OpenCV functions *cvGetTickCount()* and *cvGetTickFrequency()*.

Table 2.1: Running Time Proportions

Kernel	GPU Time(us)(%)	CPU Time(us)(%)
Image Warping	67.1 (7.9%)	6942.4 (23.6%)
Gradient Calculation	55.5 (6.5%)	1552.1 (5.3%)
Jesm Calculation	56.2 (6.6%)	4483.9 (15.2%)
Difference Calculation	18.7 (2.2%)	1921.68 (6.5%)
Equation Solving	408.7 (48.0%)	11692.5 (39.7%)
Pose Update	19.6 (2.3%)	4.4 (0.01%)
Correlation Calculation	226.3 (26.5%)	2853.9 (9.69%)
Total	852.1 (100%)	29450.88 (100%)

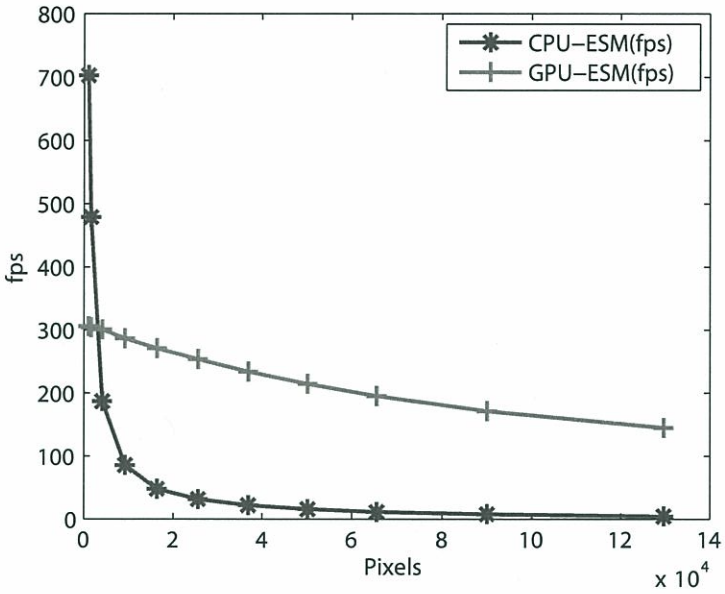
2.5.2 Experiment II: Speed Comparison of GPU-ESM and CPU-ESM

To compare the performance of our GPU-ESM with CPU-ESM, we use the same input image sequence on the same desktop. The image sequence is taken from the camera with the test image "Graffiti" by the *Robot Research Group* in University of Oxford. The image is printed out and attached on a board as a planar object for visual tracking (Figure 2.17). By counting the processing time of each image, their processing frame rates (expressed in frames per second (FPS)) are shown in Figure 2.13 (a). Their fps ratio is shown in Figure 2.13 (b).

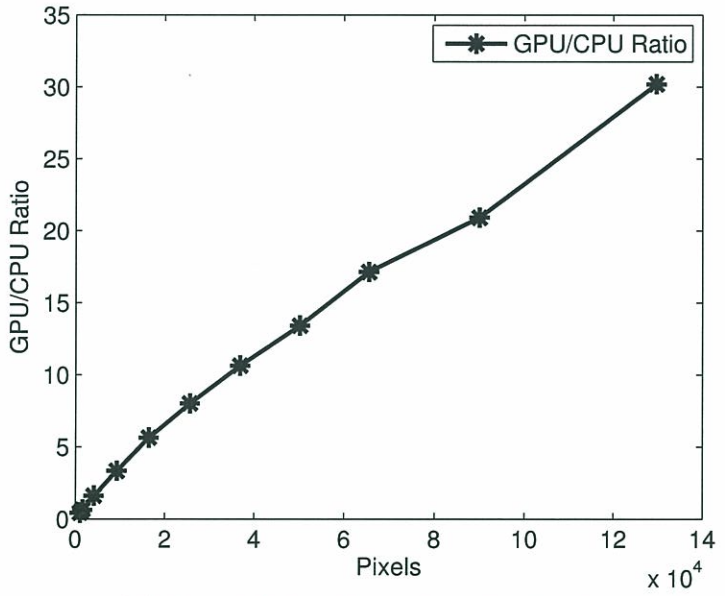
Figure 2.13 (a) shows that using GPU can greatly accelerate the ESM algorithm. Though both processing speeds decrease with the increase of tracking region size, the GPU-ESM can still work at a relative fast speed even with a large tracking region. Meanwhile, as the 'GPU/CPU Ratio' increases with the pixel number in the tracking region in Figure 2.13 (b), it also shows that a GPU is more preferable for highly parallel processing. For those algorithms which can not be parallelized, using a GPU might induce an even lower speed due to more transportation between various memories.

Figure 2.12: The test image "Graffiti" by the *Robot Research Group* in University of Oxford is printed out and attached on a board as a planar object for visual tracking.

Besides, a sample sequence of input images taken from the GPU thread is shown in 2.14. Several key frames in the GPU-ESM and the CPU-ESM tracking process are shown separately in Figure 2.15 and Figure 2.16. The tracking region is a 200×200 window shown in $t = 0.00$s. The boxed regions in the first row of Figure 2.15 and Figure 2.16 are warped back with homography and shown in their second rows. Despite illumination change, the warped regions should be very close to the reference template when the tracking is accurately performed. During the experiments, we start to move the object from $t = 5.00$s. From the sequences in Figure 2.16 the CPU-ESM performs poor with a moving object (for $t > 6.00$s the warped regions are totally different from the warped region of $t = 0.00$s) while the GPU-ESM can still perform visual tracking well (The warped regions in Figure 2.15 are nearly the same).



(a) Processing Speed Respecting to Pixels



(b) Speed Ratio Respecting to Pixels

Figure 2.13: Processing speed comparison of GPU-ESM and CPU-ESM. (a): Processing speed of GPU-ESM and CPU-ESM with different tracking region (Variables in X axis are the total pixel numbers of the tracking region). (b): The speed ratio of GPU-ESM/CPU-ESM with the same data as in (a).

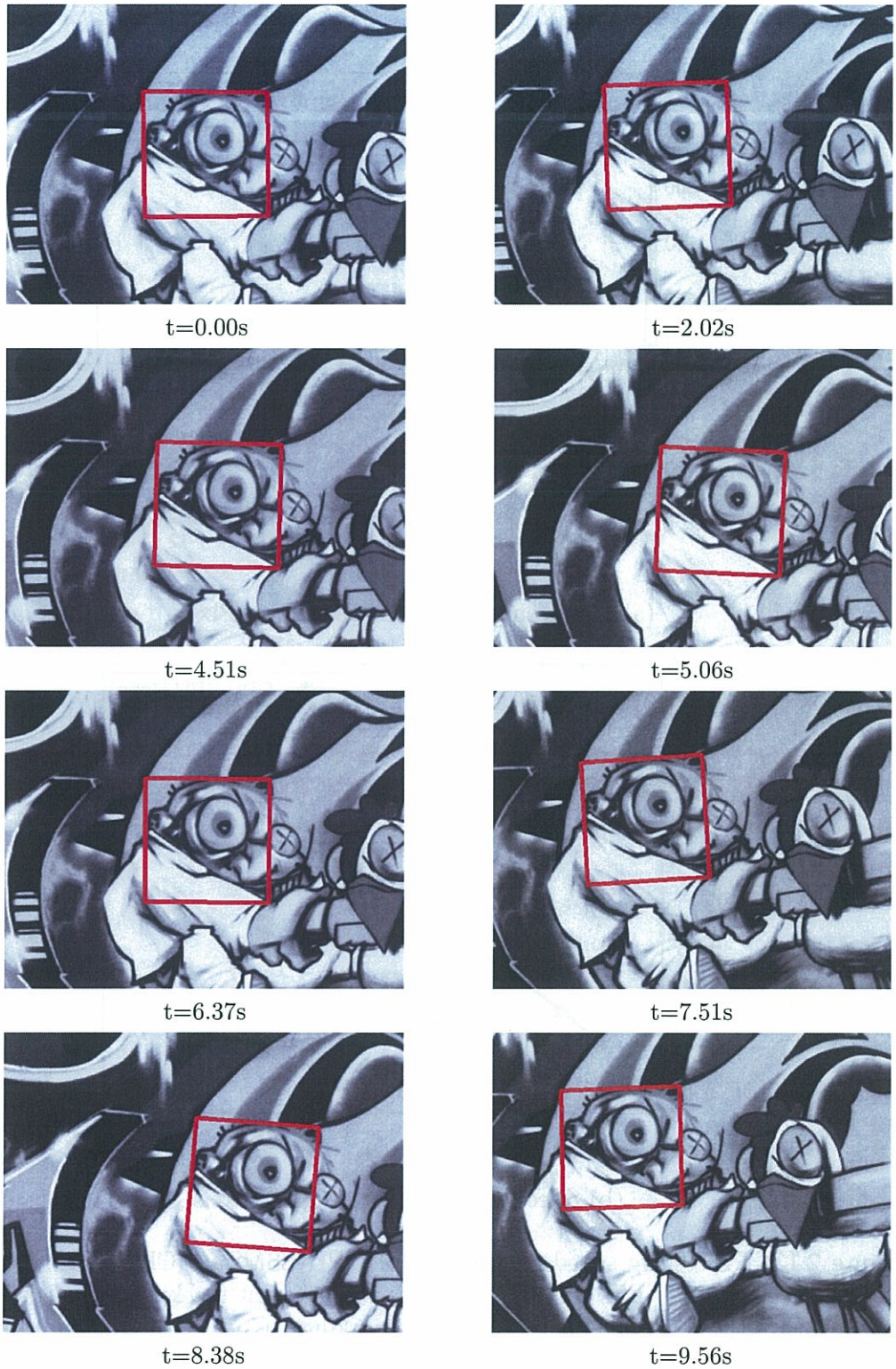


Figure 2.14: A sample sequence of input images is taken from the GPU thread. The red rectangular in the image denotes for the tracked region of GPU-ESM algorithm.

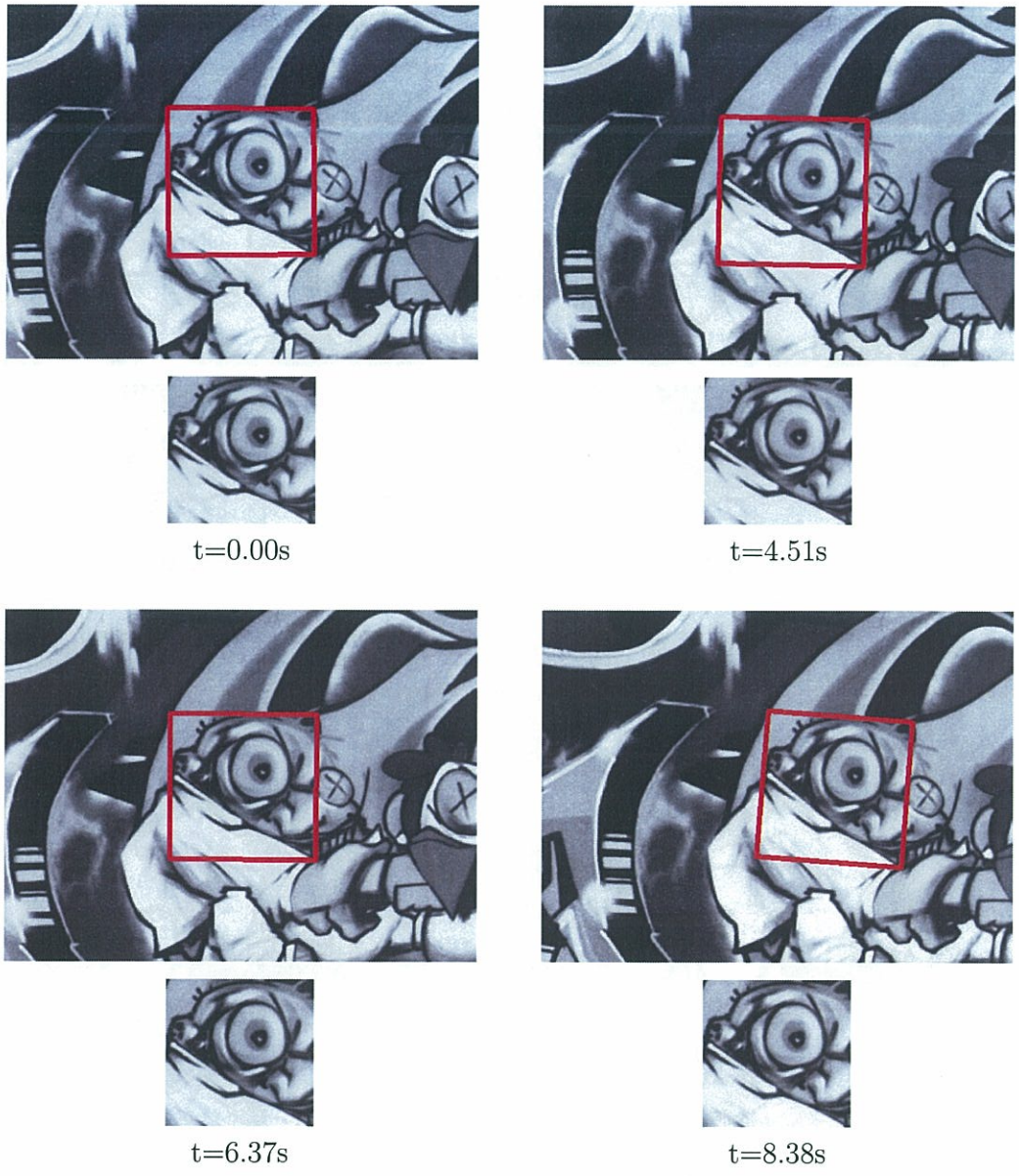


Figure 2.15: GPU-ESM. The second row shows the warped images from the boxed region in the current images (the first row). The result that all warped images are nearly the same shows that the GPU-ESM can track the fast moving object. The input images are taken from the test dataset "Graffiti" by the *Robot Research Group* in University of Oxford.

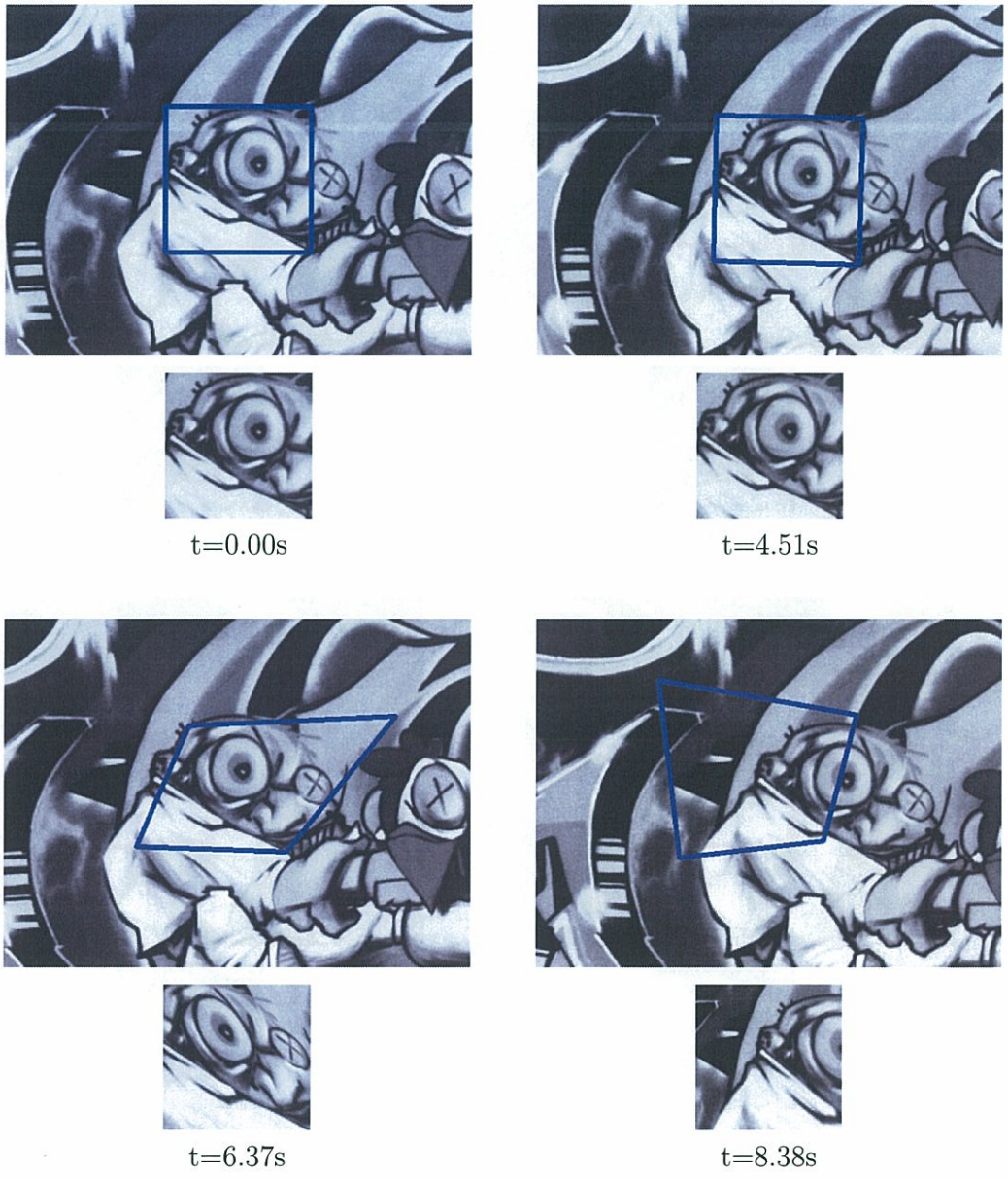


Figure 2.16: CPU-ESM. The change of warped images shows that the CPU-ESM can not track the same moving object as in Figure 2.15. The input images are taken from the test dataset "Graffiti" by the *Robot Research Group* in University of Oxford.

Figure 2.17: The test image "Graffiti" by the *Robot Research Group* in University of Oxford is attached on the box as a 3D object for visual tracking.

2.5.3 Experiment III: GPU Acceleration in Tracking Multiple Planes

In this experiment, we extend the homography-based visual tracking to a 3D tracking region based on multiple planes tracking. In many applications a 3D tracking region is composed of two or more adjacent planar regions [133]. We choose a box as the tracking object. The test image "Graffiti" by the *Robot Research Group* in University of Oxford is printed out and attached on the two joint planes as the 3D object for visual tracking. After extracting the boundary of each planar region from the reference image, the 3D region is separated into two planar regions. Then we carry out the GPU-ESM tracking algorithm on each planar region and merge the warped regions again. A sample sequence of the input images is shown in 2.18. Several key frames in the GPU-ESM tracking process are shown in Figure 2.19. Thanks to the GPU speedup, our system can track multiple planar regions at a high speed. As shown in Figure 2.19, for a tracking area of 364×208 pixels with two planar regions, the processing speed is 77 fps.

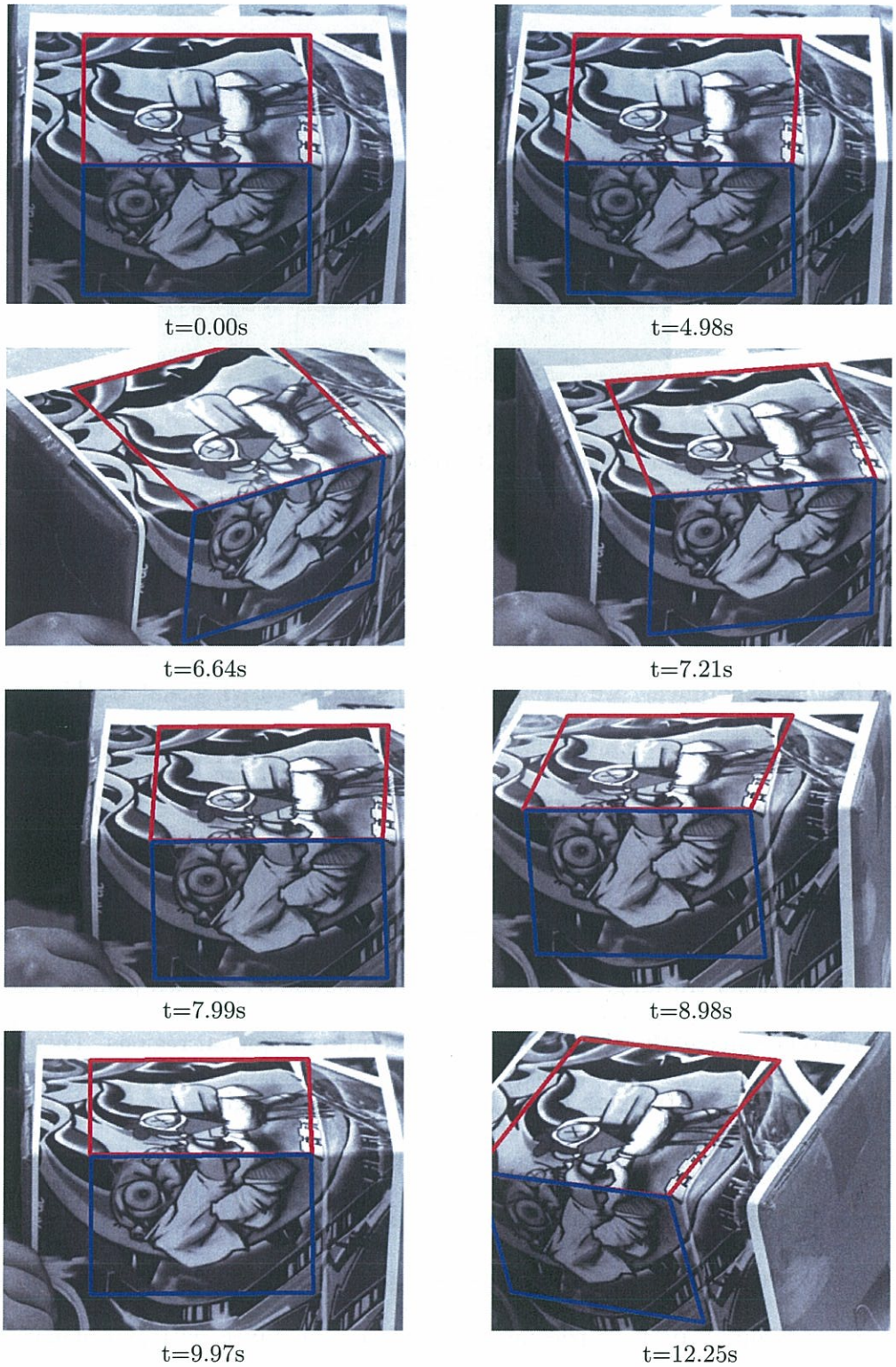


Figure 2.18: A sample sequence of input images. The red and blue rectangular in the image denotes for the two tracked regions of GPU-ESM algorithm.

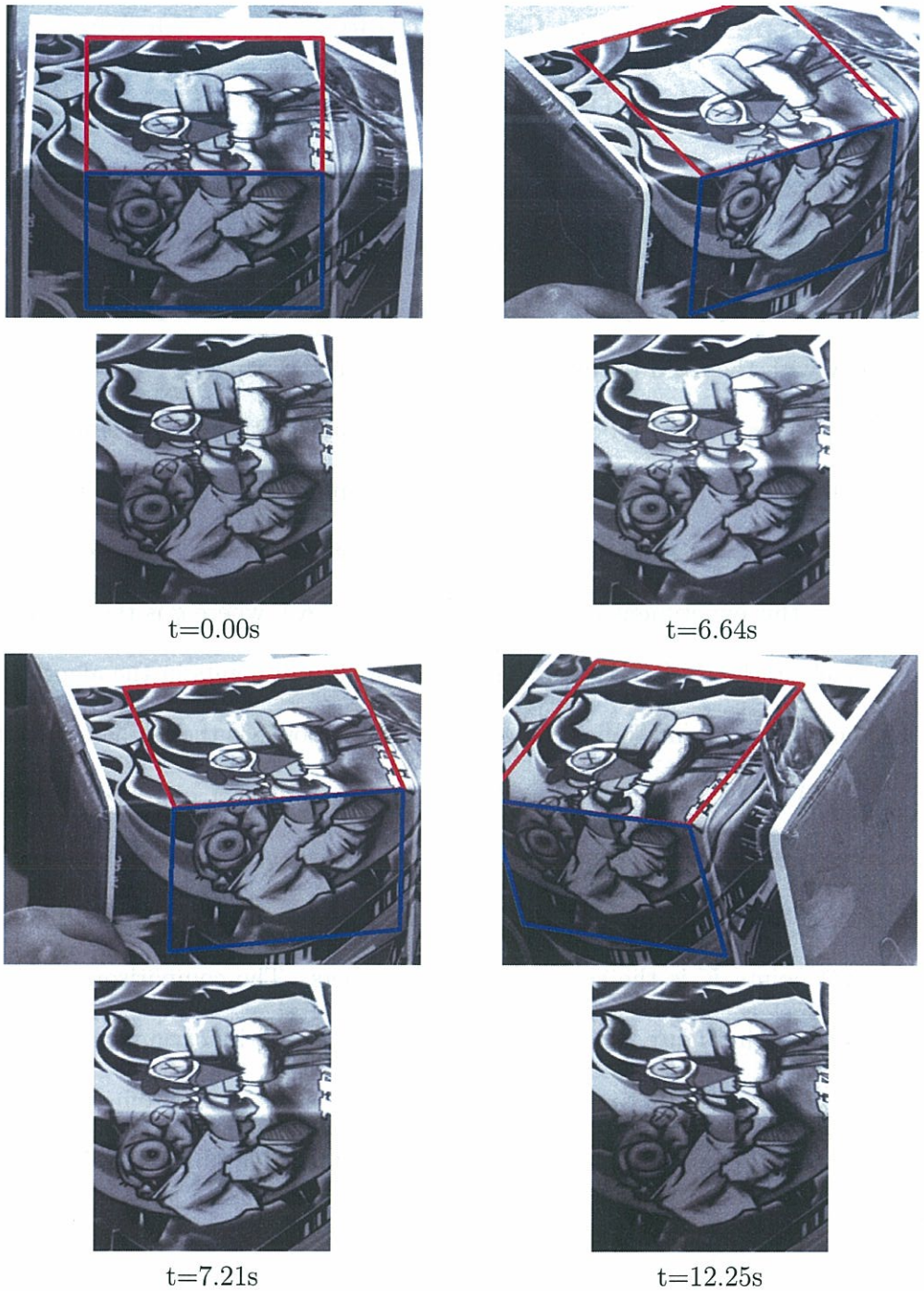


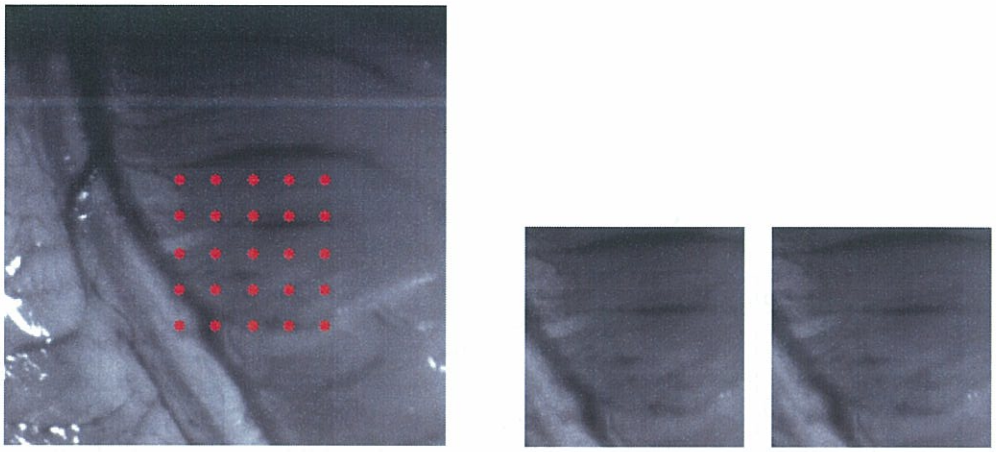
Figure 2.19: 3D object tracking based on 2 planar regions tracking. The tracking region is in two adjacent surfaces of a 3D object. Two tracking regions are described in black and white boxes separately. The warped images in the second row show that our system can track the 3D region well. The input images are taken from the test dataset "Graffiti" by the *Robot Research Group* in University of Oxford.

2.5.4 Experiment IV: GPU Acceleration in Deformable Image Registration

In this experiments we have extended our GPU acceleration technique to a similar computer vision problem, the deformable image registration with the Thin Plate Spline (TPS) proposed by Malis [132]. This method has been proved to have high efficiency with the ESM optimization algorithm in the paper.

The process flow of TPS based deformable object tracking is similar with the homography based visual tracking with ESM otpimization in 2.3.2. The difference only lies in the dimension of the $\boldsymbol{J}_{esm}$ matrix. Contrasted with the $q \times 8$ parameters in 2.3.2, in this experiment it becomes $q \times (6 + 2 \times K)$, where q is the number of the total pixels in the ROI and K is the number of control points in the algorithm.

Then the matrix multiplication result $\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}$ now has the variable dimension of $(6 + 2 \times K) \times (6 + 2 \times K)$. In the experiment, the ROI is set to 128×128 pixel from the 256×256 pixels input images. The number of control points in the algorithm has been set to $\{1, 3, 5, 7, 9, 11, 13\}$ in u and v direction respectively. The number of control points K is the square of these numbers. The comparison of processing speed is shown in Figure 2.21. We also measured the multiplication time in GPU and CPU applications. The running time is shown in Table 2.2. From the experiment the efficiency of our GPU acceleration in the deformable object tracking has been evaluated. One image sequence of the tracking the beating heart is shown in Figure 2.20. The red points are the control points used in the algorithm.



(a) Current image (b) Template (c) Warped Image

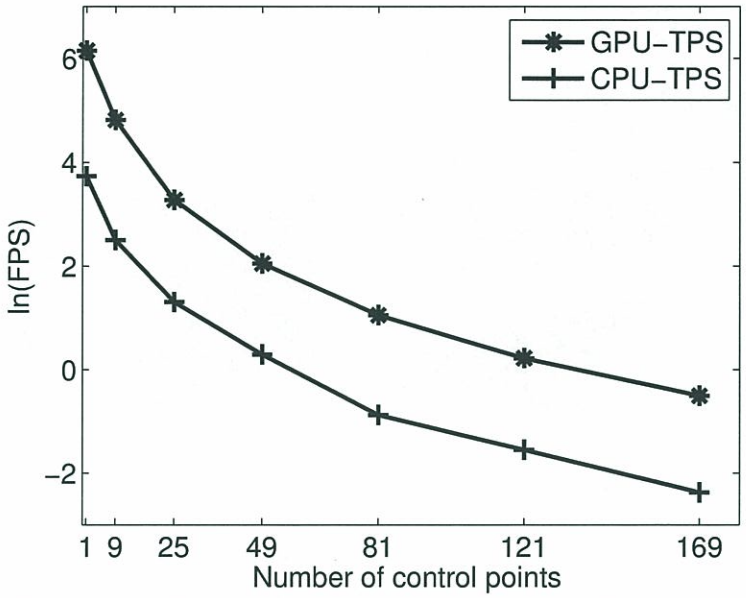
Figure 2.20: TPS based deformable image tracking with GPU acceleration. Input image is an image of beating heart of a pig. It is of 256×256 pixels and the tracking region is chosen to 128×128 pixels. The red points denotes the control points in the experiments. The input image is provided by the *Team Automatique Vision et Robotique* in University of Strasbough.

Table 2.2: Running Time Comparison (Unit: us)

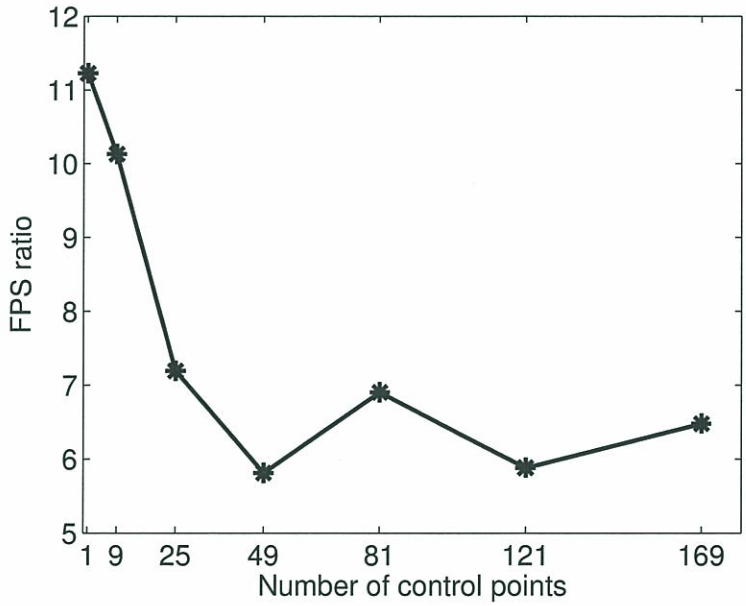
Function / K	$\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}$(GPU)	$\boldsymbol{J}_{esm}^{\top}\boldsymbol{y}$(GPU)	$\boldsymbol{J}_{esm}^{\top}\boldsymbol{J}_{esm}$(CPU)	$\boldsymbol{J}_{esm}^{\top}\boldsymbol{y}$(CPU)
1	417	111	678	123
9	1338	107	6859	366
25	7682	148	31667	975
49	23706	223	101205	1826
81	60533	1434	393666	3038
121	127383	457	789579	4235
169	244215	579	1618535	6186

Table 2.3: Comparison of Processing Speed (FPS)

Method / Number of control points K	1	9	25	49	81	121	169
GPU-TPS	470.3	123.9	26.6	7.8	2.9	1.3	0.6
CPU-TPS	41.9	12.2	3.7	1.3	0.4	0.2	0.1



(a) Processing Speed Respecting to Number of Control Points



(b) Speed Ratio Respecting to Number of Control Points

Figure 2.21: Processing speed comparison of TPS based deformable image registration in GPU and CPU applications. (a): Processing speed comparison with different number of control points (variables in X axis are the number of K). The FPS data has been processed by natural logarithm to show the data clearly. (b): The speed ratio of both applications with the same data as in (a).

2.6 Conclusions

In this chapter, the implementation of the GPU-ESM algorithm is presented. By utilizing optimization techniques such as memory optimization and memory coalescing and strided access, the GPU-ESM algorithm provides us a better system performance with a higher processing speed. The experimental results evaluate the efficiency and effectiveness of our GPU applications. By investigating the optimization techniques adopted in our implementations in detail, this chapter also makes contribution to the general purpose GPU computation community.

Chapter 3

Camera Positioning by Homography-based Visual Servo

This chapter concentrates on solving the camera positioning task with respect to a planar object. We adopt the homography-based visual servo to move the camera to its desired pose. We have proposed a combination strategy of GPU-ESM tracking algorithm and GPU-SIFT matching algorithm for homography estimation. With this combination strategy, the system can provide fast and accurate homography solution from a large initial differences range. This chapter describes the combination strategy in detail. The implementation of the combination strategy and the visual servo scheme is introduced. Experiments results are shown to evaluate the efficiency of these algorithms.

3.1 Introduction

3.1.1 Homography-based Visual Servo

In this chapter we concentrate on solving the camera position task with respect to a planar object. As an important property in the images of planar objects, homography has been used in visual servo to realize various applications. A survey of homography-based visual servo is shown in section 1.2.2 from page 14. In these methods, Benhimane and Malis have proposed a homography-based visual servo scheme [47], in which the homography solution is directly adopted in the control law. Consequently, neither homography decomposition nor 3D parameter estimation is needed. Therefore, we adopt their homography-based visual servo scheme to place a camera with respect to 2D planar objects.

As the homography is directly used in this control law, the estimation of homography becomes the key problem. In chapter 2 we have proposed a GPU accelerated ESM tracking algorithm to provide fast homography. However, when applied in a real visual servo tasks, there are still several obstacles. One limit is the small convergence range due to the local optimization nature of the ESM tracking algorithm for homography solution. It can not find the right solution if the initial homography are far from the real one. However, in real applications there might be large differences between the initial pose and the desired pose. This camera pose difference causes large differences between the initial image and the template image. The ESM algorithm can not find an accurate solution for the reason of too small overlapping region between the two images. Furthermore, when the current homography estimation is not accurate, such as in some extreme cases when partial occlusion happens, the ESM

algorithm can not continue tracking the region with the bad previous homography due to its local optimization nature.

Therefore, the key problems consists in finding a good initial condition to initialize the ESM tracking algorithm, that is, finding the homography within large initial pose ranges and after the occlusion happens. As the homography can be estimated from the corresponding pairs of image points in the two image, image matching algorithms can help to find a homography solution from two images with large differences. Lowe proposed the Scale Invariant Feature Transform (SIFT [134]) algorithm to match two images with the SIFT local features. The SIFT matching method has been demonstrated to have high accuracy with respect to variations in scale, rotation, and translation. Accordingly, we adopt the SIFT algorithm to realize the image matching in those extreme cases for the ESM algorithm. Then the homography can be estimated from the matched image features.

3.1.2 Goals and Contributions

In this chapter our goal is to realize the camera positioning with respect to planar objects. Our contributions are mainly as follows.

We have proposed a combination strategy for homography estimation so that it can be used in the homography-based visual servo to place a camera with 2D planar objects. The combination strategy is composed of the ESM tracking algorithm in chapter 2 and the GPU accelerated SIFT (GPU-SIFT) algorithm, which has been extended with RANdom SAmple Consensus (RANSAC) to estimate the homography.

In this combination strategy, the ESM tracking algorithm provides fast homography estimation, meanwhile the initial homography, which is the key issue for the

ESM tracking algorithm as a local optimization method, is provided by the SIFT matching algorithm. With this strategy, when the ESM tracking algorithm fails due to large initial error or occlusions, the homography from the SIFT matching algorithm is loaded to restart the ESM tracking. By this means, the initial pose range has been enlarged and the system can continue placing the camera after occlusion happens.

We have adopted the multi-thread programming technique to run both GPU-ESM and GPU-SIFT algorithms on separate GPUs simultaneously to ensure high efficiency. We have proposed a switch strategy between both methods based on the ZNCC value when occlusion happens. With the proposed combination strategy, we have improved the system performances of processing speed, convergence range and the robustness to occlusions.

3.1.3 Outline of the Rest Parts

The rest of this chapter is organized as follows. Section 3.2 describes the algorithms adopted in our system. Section 3.3 introduces the implementation details of the GPU-SIFT algorithms and the combination strategy. Section 3.4 describes the experimental results to evaluate our combination strategy. Section 3.5 concludes this chapter.

3.2 Related Algorithms

3.2.1 SIFT Matching Algorithm for Homography Estimation

Scale Invariant Feature Transform (SIFT) was proposed by Lowe in 1999 [134]. As an algorithm to detect and describe local features in the images, it has been demonstrated to have high accuracy with respect to variations in scale, rotation, and translation. With the matched SIFT feature pairs in the two images, we have extended the SIFT features to estimating the homography between these two images. The processing loop for an input image is as follows.

1. **Feature Detection**. The SIFT features from the current image and the reference image are extracted respectively. Here the current image means the image taken from the initial camera pose, while the reference image means the image taken from the desired pose. The searching region of the feature points is the whole image to ensure a global match. The computation of SIFT feature detection from the reference image is computed in advance as the feature points stay the same during the whole servo process.

2. **Feature Points Matching**. From the detected SIFT feature points in the reference image and the current image, a SIFT matching process is carried out to match the corresponding pairs of feature points between the two images.

3. **RANSAC for Homography**. After finding the matched feature points, the RANSAC method is used to estimate the homography in the corresponding pairs of feature points. The RANSAC method has shown better performances than the ordinary least squares methods as it can effectively remove the outliers (the mismatched pairs of points in SIFT algorithm).

3.2.2 Homography-base Visual Servo

Given a known homography $\boldsymbol{G}$ between two images of a planar object and the camera's intrinsic parameter matrix $\boldsymbol{K}$, a matrix $\boldsymbol{H}$ can be calculated as

$$\boldsymbol{H} = \boldsymbol{K}^{-1}\boldsymbol{G}\boldsymbol{K}. \tag{3.1}$$

With $\boldsymbol{H}$ the homography-base visual servo task function $\boldsymbol{e}$ in Benhimane's method [47] is defined as

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{e}_{\upsilon} \\ \boldsymbol{e}_{\omega} \end{bmatrix}, \tag{3.2}$$

where

$$\begin{aligned} \boldsymbol{e}_{\upsilon} &= (\boldsymbol{H} - \boldsymbol{I})m^*, \\ [\boldsymbol{e}_{\omega}]_{\chi} &= \boldsymbol{H} - \boldsymbol{H}^{\top}. \end{aligned} \tag{3.3}$$

The m^* is the center of gravity of the region. The $[\boldsymbol{e}_{\omega}]_{\chi}$ is the skew-symmetric matrix[1] derived from the vector $\boldsymbol{e}_{\omega}$. The visual servo control law is

$$\begin{bmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{bmatrix} = - \begin{bmatrix} \lambda_{\upsilon}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_{\omega}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_{\upsilon} \\ \boldsymbol{e}_{\omega} \end{bmatrix}, \tag{3.4}$$

where $\boldsymbol{\upsilon}$ and $\boldsymbol{\omega}$ are the camera translation and rotation velocity respectively in the camera frame, $\lambda_{\upsilon} > 0$, $\lambda_{\omega} > 0$ are constant gain.

To be mentioned, in this visual servo scheme, the computation of the task function and the control law only depends on the matrix $\boldsymbol{H}$. No other parameter estimation is needed.

[1] The relationship between a 3×1 vector and a 3×1 screw-symmetric matrix is (B.8) on page 163.

Table 3.1: Comparison of Different Methods

Method	Speed	Searching Area	Occlusion
CPU-ESM	Slow	Local	Vulnerable
GPU-ESM	Fast	Local	Vulnerable
GPU-SIFT	Slow	Global	Robust
GPU-ESM+ GPU-SIFT	Fast	Global	Robust

3.2.3 Comparison of Different Methods

Until now we have presented the ESM tracking algorithm and SIFT matching algorithm for homography estimation. The comparison of different methods is shown in Table 3.1. As mentioned before, the GPU-ESM tracking algorithm can provide a fast and accurate homography when the solution is near the global minimum point, i.e. the image difference is small. The limit of the ESM method is its small convergence region (local). Furthermore, the ESM tracking algorithm is vulnerable to partially occlusions. Meanwhile, by matching the SIFT feature points between the two whole images, the SIFT algorithm with RANSAC extension can offer a robust homography solution in a large region or in the occlusion cases. After the occlusion is moved, the SIFT matching algorithm can continue providing a reliable homography solution. But compared with GPU-ESM, it is slow even with its GPU accelerated version (GPU-SIFT). Therefore it is necessary to make an effective combination of GPU-ESM and GPU-SIFT to ensure a fast and robust system. The combination implementation is shown in detail in section 3.3.2.

3.3 Implementation of Camera Positioning

This section describes the implementation of the homography-based visual servo framework for camera positioning. The GPU accelerated SIFT algorithm and the

combination strategy are introduced. The combination of GPU-ESM and GPU-SIFT algorithm and the multi-thread programming model are presented in detail.

3.3.1 GPU Accelerated SIFT Algorithm

As one kind of local descriptor features in image processing, SIFT algorithm has been demonstrated to have high accuracy with respect to variations in scale, rotation, and translation. However, its computation involves a 128 dimensional descriptor which is computationally intensive and difficult to apply for a realtime application. To accelerate its processing speed, various algorithms have been proposed, including PCA-SIFT [102] and SURF (Speed up Robust Features) [105]. And there are also hardware accelerated SIFT such as GPU SIFT implementations [135], multi-core accelerated SIFT [136], and so on. In our system we adopt the GPU implementation 'SiftGPU' by C. Wu [137]. By exploiting the processing power of a GPU, we have achieved a significant speedup (20 times) on the SIFT feature extraction and matching over the CPU implementation.

3.3.2 Implementation of the Combination Strategy

To build a stable and efficient system, we combine the GPU-ESM and GPU-SIFT to improve the system performances. The combination model is shown in Figure 3.1. In our system, both GPU-ESM and GPU-SIFT algorithms run on separate GPUs simultaneously to process the input images.

In the GPU-ESM algorithm, after processing each image in one loop, the ZNCC correlation value is checked to determine whether the ESM algorithm fails or not. If tracking failure happens, the GPU-ESM will automatically load the current homog-

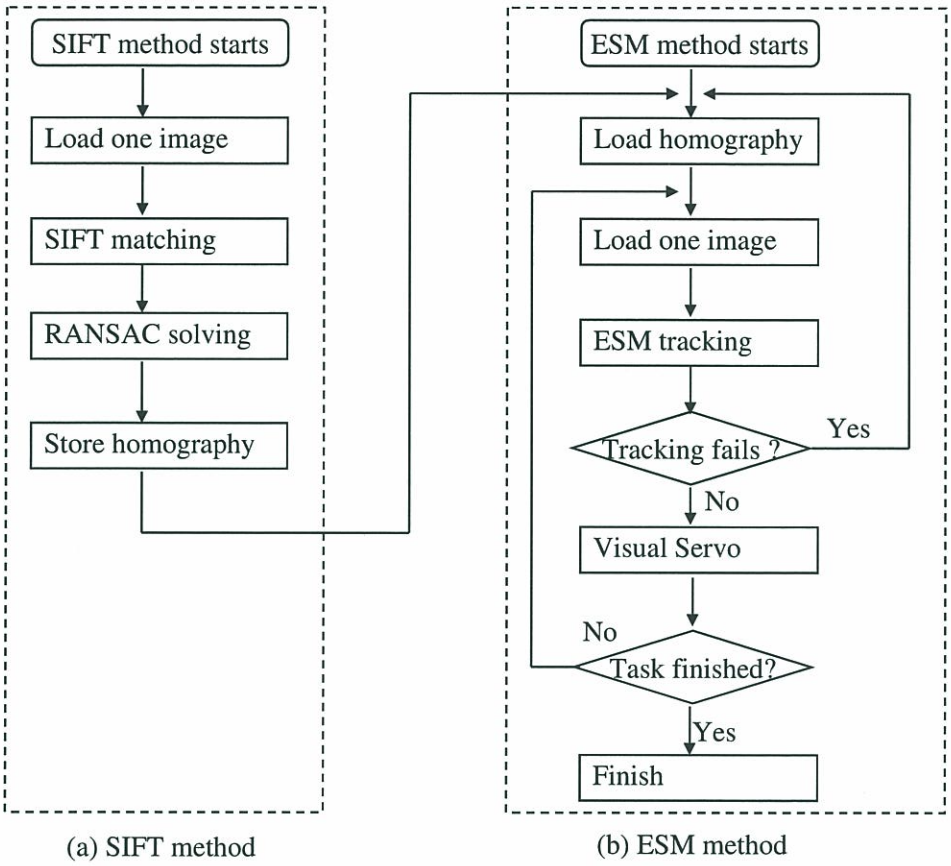


Figure 3.1: Combination of the GPU-ESM and the GPU-SIFT algorithms for homography estimation.

raphy solution from the GPU-SIFT and set it as the new initial value of the iterative ESM processing loop. By this means, the GPU-ESM algorithm can continue working and providing a stable and highly accurate homography solution. Therefore, the whole homography-based visual servo system can work smoothly with high reliability at a high processing speed.

We adopt the multi-thread programming technique in this combination (Figure 3.2). Three independent threads are created for the image capture, GPU-ESM and GPU-SIFT respectively. Three blocks of shared memories are created to share the image data between the capture thread and each processing thread. Semaphores are

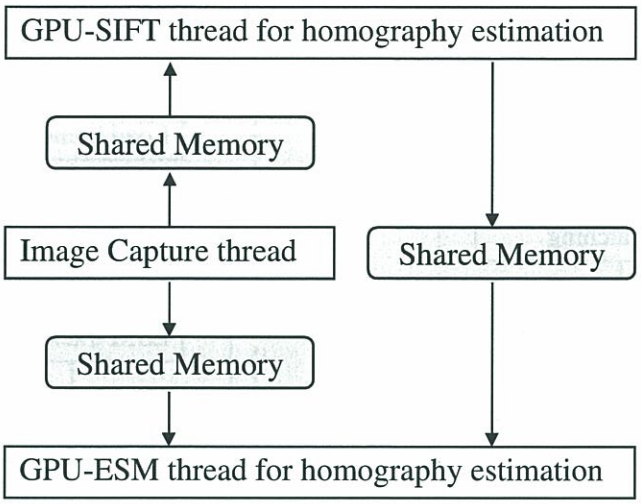


Figure 3.2: Multi-thread programming model. Three independent threads are created for the Image Capture, GPU-ESM and GPU-SIFT respectively. Three blocks of shared memories are allocated to share the data between three threads.

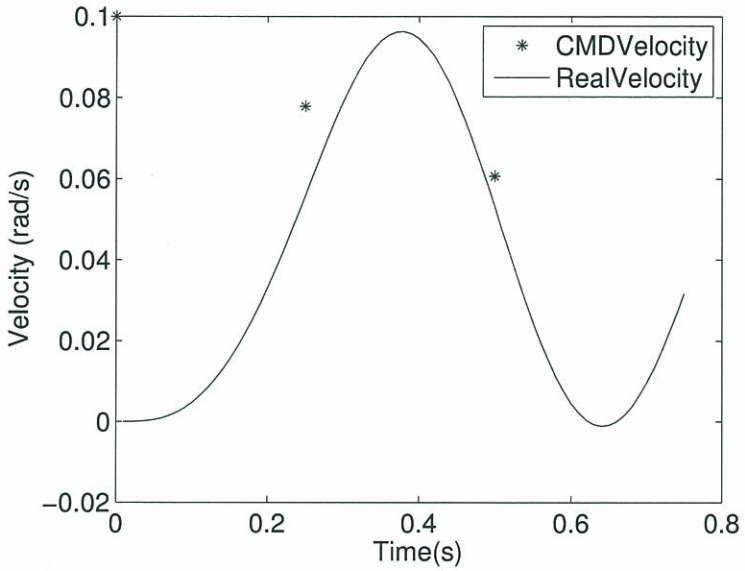
used to control the access of the shared memories between these threads.

After the image capture thread capturing one image, this image data will be send to both shared memories of GPU-ESM and GPU-SIFT threads with its own index. If the image in the shared memory has not been processed when the new image comes, the old image will be deserted and the new image will be stored. Therefore, strictly speaking, at an instantaneous both GPU threads might be processing two images with different indexes due to the different processing speed of GPU-ESM and GPU-SIFT algorithms. Such as, after the occlusion happens in the $N\ th$ frame, the GPU-ESM will load a homography from the GPU-SIFT, which might be computed from a differnt frame $N-1$, or $N-2$ or even other frames. But due to the high framerate of the camera (200 fps), there will be little displacement between these frames. Therefore, the homography from the GPU-SIFT can be used to initialize the GPU-ESM algorithm and the GPU-ESM can continue estimating the homography.

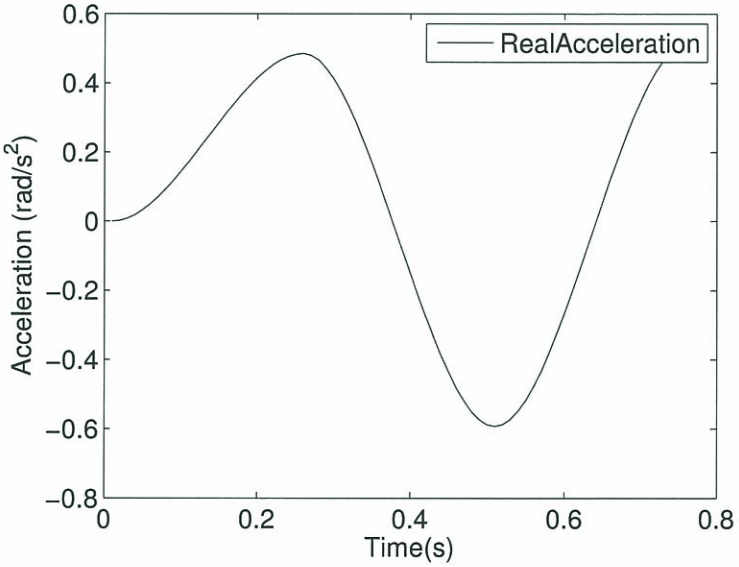
3.3.3 Velocity Interpolation

The output of the visual servo control is the velocity screw with respect to the camera frame $\sum_c$, that is ${}^c\boldsymbol{V}_{6\times1} = [\boldsymbol{v}\ \boldsymbol{\omega}]^\top$. Therefore, the transformation from camera frame $\sum_c$ to the robot base frame $\sum_0$ in Figure C.2 is required to move the robot to fulfill such camera velocity demand. Velocity transformation from ${}^c\boldsymbol{V}$ to ${}^0\boldsymbol{V}$ (velocity expressed in the robot base frame $\sum_0$) is described in Appendix D in detail. Given this velocity ${}^0\boldsymbol{V}$, the robot Jacobian matrix $\boldsymbol{J}$ is used to determine the desired joint velocities $\dot{\boldsymbol{\theta}} = [\dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5]$ of the robot by (D.5).

However, to move the robot fast and smoothly in the homography-based visual servo system, several practical techniques have to be considered. The velocity command from the visual servo control law is discrete (with period T) but the real joint motion is with a even smaller sampling period. And also in the initial case, due to the large initial error, a large velocity will be calculated from the visual servo control law, meanwhile the robot joint motors are static in the beginning. Therefore, this discontinuity of velocity will induce undesirable vibrations on the robot and may cause early wear and tear of the mechanical parts of the robot. For high speed robots, it is worth ensuring the continuity of acceleration as well in order to avoid exciting resonances in the mechanics. Consequently, we adopt the polynomial interpolation with velocities to move the robot fast and smoothly. A sample of velocity interpolation is shown in Figure 3.3.



(a) Velocity Interpolation



(b) Acceleration

Figure 3.3: A sample figure of velocity interpolation. (a): The velocity command from visual servo is discrete and drawn with star markers. The velocity after polynomial interpolation are shown with a continuous curve. At $t = 0$ the velocity command from visual servo controller is large due to the large initial image difference, meanwhile the motor velocity is 0. (b): The acceleration continuity is also kept in the polynomial interpolation.

Here we take an interpolation task from velocity V_1 to velocity V_2 in period T to illustrate our strategy. To keep the continuity in velocity and acceleration, the velocity interpolation is realized as

$$V(t) = a + bt + ct^2 + dt^3 + et^4,\ t \in [0, T], \tag{3.5}$$

with boundary conditions for velocity and acceleration continuity:

$$\begin{aligned} &V(0) = V_1;\ V(T) = V_2; \\ &A(0) = A; \\ &A'(0) = 0;\ A'(T) = 0. \end{aligned} \tag{3.6}$$

From these five boundary conditions, we can solve the five coefficients and get the final velocity interpolation as

$$V(t) = V_1 + At + \frac{2}{T^3}(V_2 - V_1 - AT)t^3 - \frac{1}{T^4}(V_2 - V_1 - AT)t^4,\ t \in [0, T]. \tag{3.7}$$

3.4 Evaluation Experiments

3.4.1 Hardware and Software Configuration

Experiments have been carried out to evaluate our proposed combination strategy. The first experiment describes the comparison of tracking accuracy of GPU-ESM and GPU-SIFT. The second one is carried out to verify the efficiency of our combination strategy. The third experiment is to verify the homography-based visual servo scheme. In the final part we present a visual inspection application with the proposed combination strategy. All experimental images are captured from a 200 fps camera (Grasshopper GRAS-03K2M/C). Size is 640 × 480. The robot is a six degrees of freedom robot (*EPSON Pro Six PS5*). The computation platform is a desktop with Intel Core i7-920 (2.67 GHz), 3GB RAM and a NVIDIA GTX295 graphic board. The GTX295 board integrates two GTX280 chips inside and has 896MB GPU RAM for each GTX280 GPU. The operating system is Windows XP (sp2). We implement the combination of GPU-ESM and GPU-SIFT algorithms with NVIDIA's CUDA (Compute Capability 1.3).

3.4.2 Experiment I: Comparison of Tracking Accuracy

In this experiments, to evaluate the tracking accuracy with GPU-ESM and GPU-SIFT, we use the test dataset from the *Robot Research Group* in the University of Oxford [127] with ground truth homography available[2]. We use the "Graffiti" dataset with six images of 800 × 600 (Figure 3.5). We have adopted two criteria to compare these two methods' accuracy.

[2]The test image dataset can be downloaded from *http://www.robots.ox.ac.uk/~vgg/data/data-aff.html.*

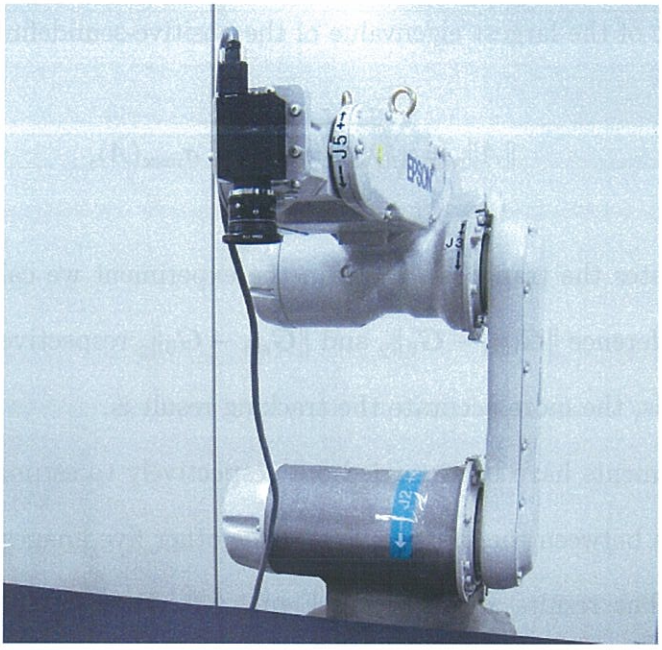


Figure 3.4: The real robot system

As introduced in Section 2.1, visual tracking consists in finding the image points of a certain interested region of an object from the current input image (Figure 2.2). Therefore, the first criterion is the ZNCC value between the template region in the reference image and the warped region in both algorithms. Here the warped region has the same size as the template region and it is warped from the new-found image region in the current image with the homography $\boldsymbol{G}$. The larger the ZNCC value is, the smaller differences the two regions have, hence the more accurate the tracking result is.

The second one is the matrix norm which is used to directly calculate the error between the estimated homography matrices $\boldsymbol{G}_{\text{esm}}$ and $\boldsymbol{G}_{\text{sift}}$ from the two methods and the ground truth homography $\boldsymbol{G}_0$ provided in the dataset. We use the spectral norm to compare the matrix difference.

The spectral norm of a matrix A is defined as the largest singular value of A, i.e.,

the square root of the largest eigenvalue of the positive-semidefinite matrix $A^\top A$:

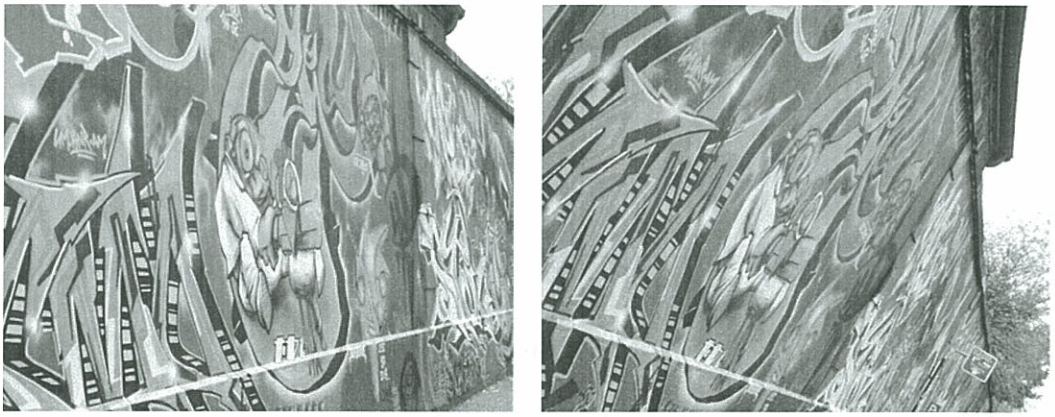
$$\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)} = \sigma_{\max}(A), \tag{3.8}$$

where $A^\top$ denotes the transpose of A. In the experiment we calculate the spectral norm of the difference $\|\boldsymbol{G}_{\text{esm}} - \boldsymbol{G}_0\|_2$ and $\|\boldsymbol{G}_{\text{sift}} - \boldsymbol{G}_0\|_2$ respectively. The smaller the spectral norm is, the more accurate the tracking result is.

Two experiments have been carried out respectively to estimate the five homography solutions between the first image and the other five images within the dataset (Figure 3.5). The results are shown in Figure 3.6. Parts of the input images and tracking results are shown in Figure 3.7 and Figure 3.8. In these experiments, both algorithms have similar accuracy on estimating the homography with tiny differences. The ZNCC values show that both the GPU-ESM tracking and GPU-SIFT matching algorithm can provide accurate homography solutions with planar objects.

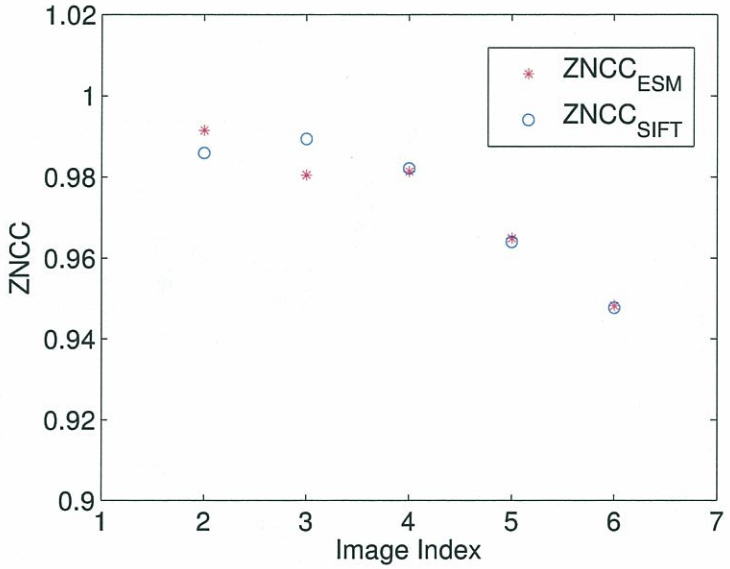
(a) Image1 (b) Image2

(c) Image3 (d) Image4

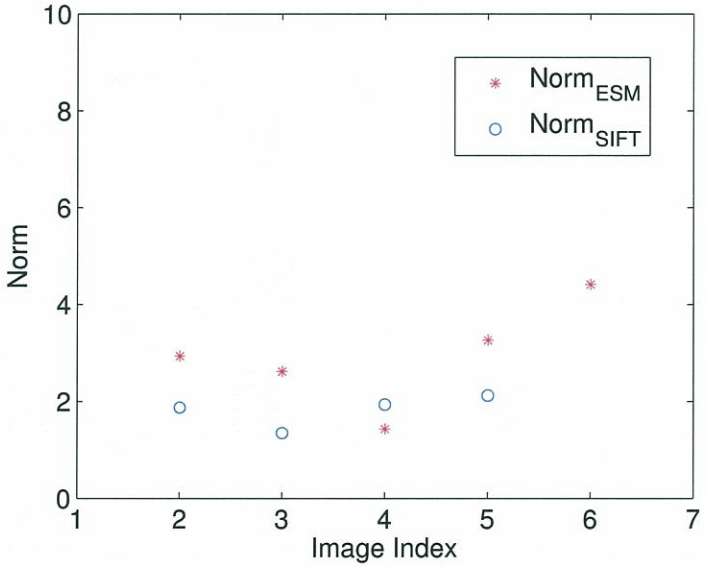


(e) Image5 (f) Image6

Figure 3.5: The set of test images "Graffiti" by the *Robot Research Group* in University of Oxford. The first image (Image1) in the dataset is used to calculate the five homography solutions between this image and the other five images.

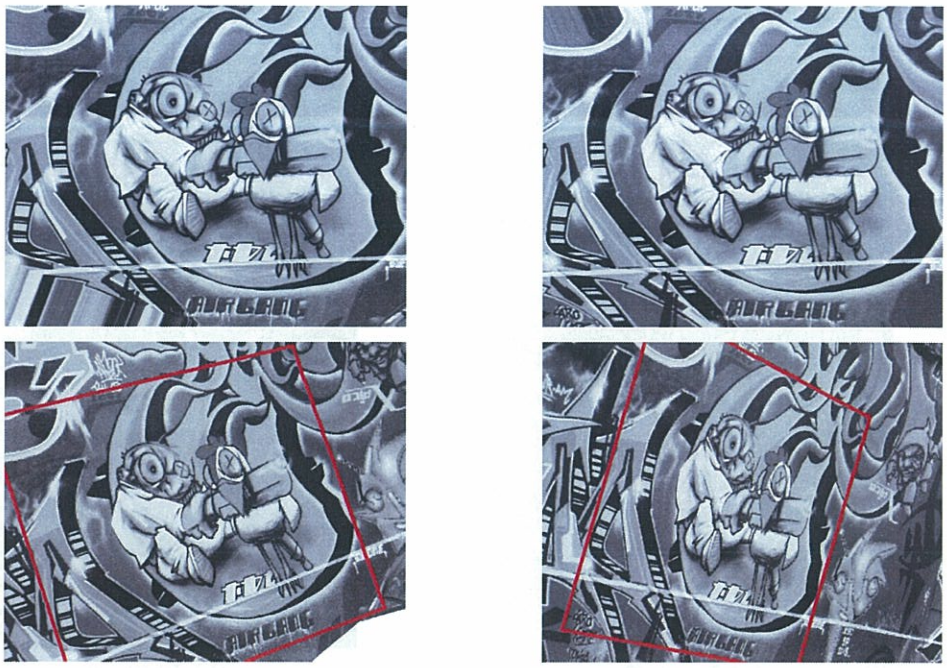


(a) ZNCC



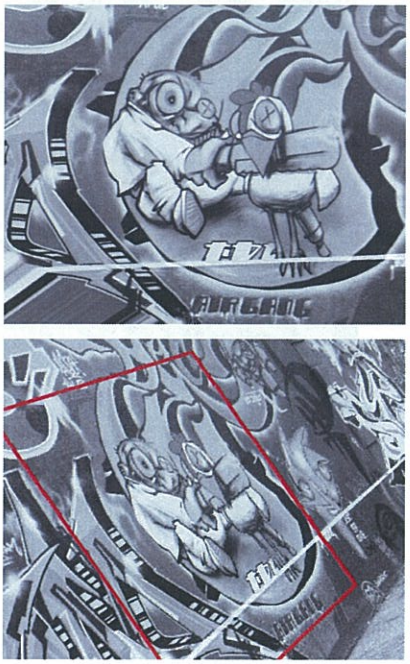
(b) Spectral norm

Figure 3.6: Accuracy comparison of GPU-ESM and GPU-SIFT. (a): The ZNCC values between the warped image (warped from the current image with the estimated homography $\boldsymbol{G}_{\text{esm}}$ and $\boldsymbol{G}_{\text{sift}}$) and the template image in both algorithms. (b): The spectral norm of the difference between the estimated homography $\boldsymbol{G}_{\text{esm}}$, $\boldsymbol{G}_{\text{sift}}$ and the ground truth homography $\boldsymbol{G}_0$.

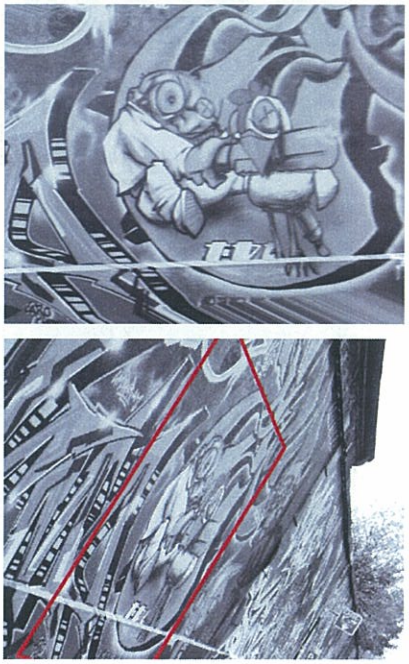


(a) Image 2

(b) Image 3



(c) Image 4



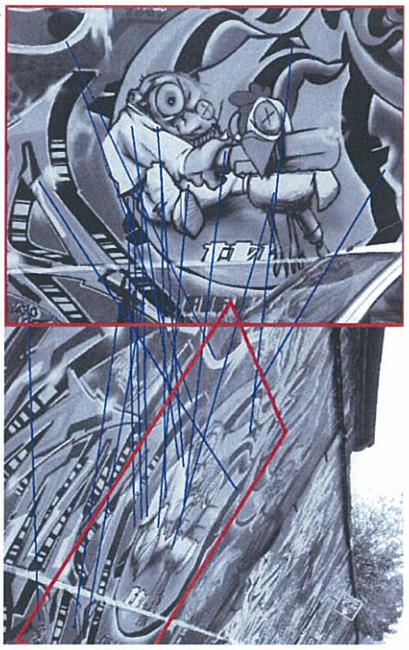
(d) Image 6

Figure 3.7: Figures of GPU-ESM tracking with parts of images in the dataset. We set the tracking target region to the whole image (800×600) pixels. ZNCC values and spectral norm values have been calculated in the experiment. The red rectangular shows the tracked region in the image.

(a) Image 2

(b) Image 3

(c) Image 4



(d) Image 6

Figure 3.8: Figures of GPU-SIFT matching with parts of images in the dataset. We set the matching area to the whole image (800 × 600 pixles). ZNCC values and spectral norm values have been calculated in the experiment. The red rectangular shows the matched region in the image. One blue line stands for a matched pair of SIFT feature points in the two images.

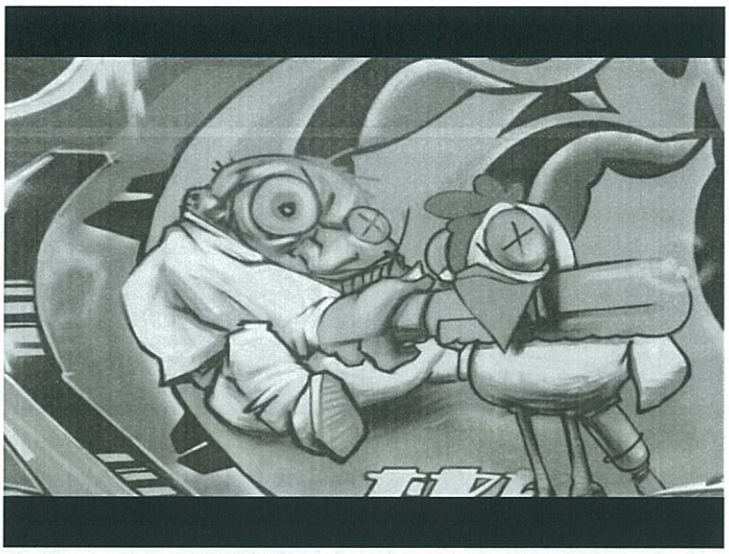
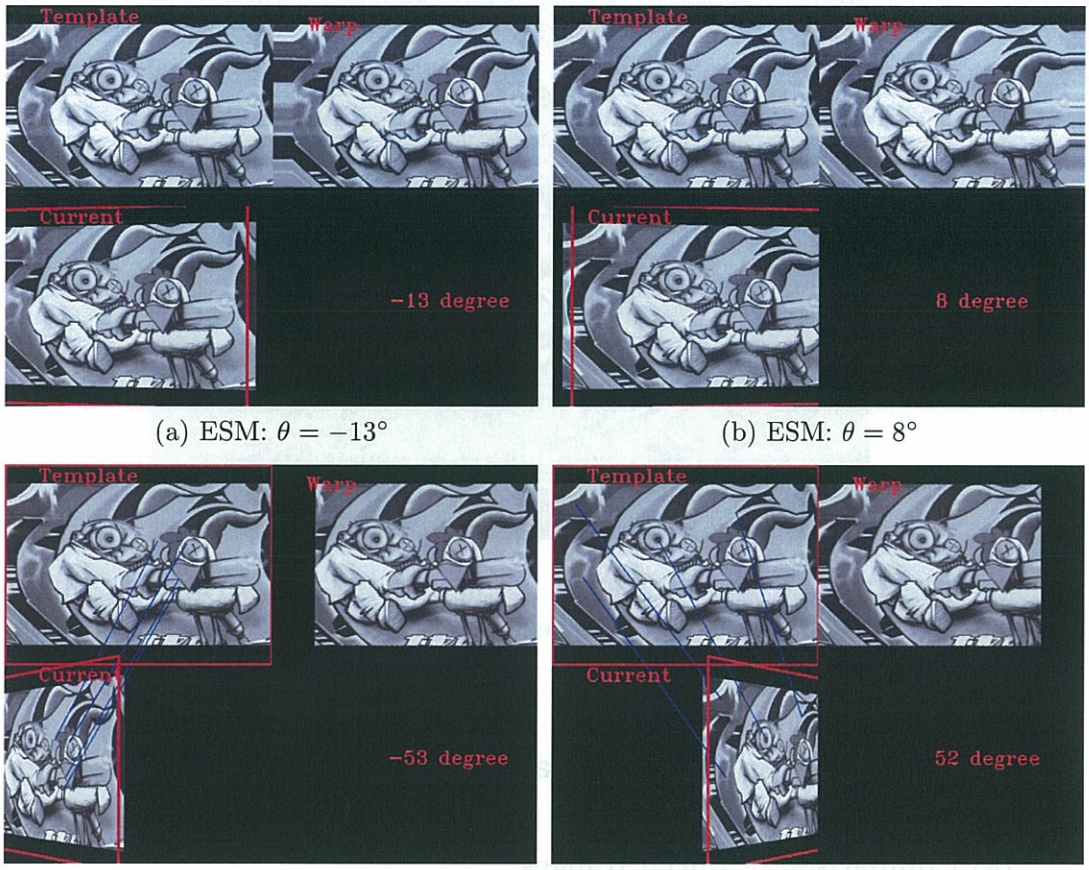


Figure 3.9: The test image "Graffiti" by the *Robot Research Group* in University of Oxford is loaded into OpenGL as texture and set as the reference image in the experiment (rotation angle is 0°).

3.4.3 Experiment II: Evaluation of Combination Strategy

(a) Comparison of Initial Ranges

Experiment is carried out to show that our combination of SIFT matching algorithm and ESM tracking algorithm has enlarged the initial range than in the ESM tracking method only. In the experiment, the test image is loaded in OpenGL as a texture with a rotation angle 0° (Figure 3.9). This image is taken as the reference image in the ESM and SIFT methods. Then the object is rotated in OpenGL about its Y axis from $-75°$ to $75°$ with step of $1°$. For each step one image is rendered and used as the current image. Then both ESM and SIFT methods try to estimate the homography between the reference image and the current image. After checking the ZNCC values in the experiment, the ESM can find accurate homography when the rotation angle θ is within $[-13°, 8°]$, while the SIFT matching method can work in the range of $[-53°, 52°]$ (Figure 3.10). This experiment result shows that SIFT



(a) ESM: $\theta = -13°$ (b) ESM: $\theta = 8°$

(c) SIFT: $\theta = -53°$ (d) SIFT: $\theta = 52°$

Figure 3.10: The object is rotated in OpenGL about its Y axis from $-75°$ to $75°$ with step of $1°$. The ESM can find accurate homography when the rotation angle θ is within $[-13°, 8°]$, while the SIFT matching method can work in the range of $[-53°, 52°]$. This has shown that our combination strategy can work in larger initial range than ESM tracking method does.

matching method can work in a larger initial range than the ESM tracking algorithm does. As our combination adopt SIFT matching method for initial pose estimation, the efficiency of our combination strategy with larger work range has been proved.

(b) Evaluation of Robustness to Occlusions

To evaluate the robustness of our combination strategy, a comparison experiment between the GPU-ESM method only and the combination of GPU-ESM and GPU-SIFT methods is carried out with occlusion cases (Figure 3.14). In the experiment,

Figure 3.11: The test image "Graffiti" by the *Robot Research Group* in University of Oxford is printed out and attached on a board as a planar object for visual tracking.

the test image "Graffiti" by the *Robot Research Group* in University of Oxford is printed out and attached on a board as a planar object for visual tracking (Figure 3.11). In the combination strategy, the lower ZNCC threshold to determine whether the ESM tracking fails or not is set to 0.6. Both methods process the same image sequence from a real camera. We add occlusions at $t = 3.61$s and $t = 6.44$s by cover the object with a white paper. The ZNCC value of the template region and the warped region is shown in Figure 3.12). A sample sequence of input images taken from the GPU-ESM thread is shown in 3.13. Several key frames in each method are shown separately in Figure 3.14, Figure 3.15 and Figure 3.16.

In the GPU-ESM only method (Figure 3.14), it can not continue tracking the region after occlusions. Instead, it converged to other local minimum solutions. Meanwhile in the combination strategy, when occlusion happens, the ZNCC value falls down rapidly and invoke the GPU-ESM to load the homography from the GPU-SIFT. After the occlusion is moved, the GPU-ESM can continue tracking the object

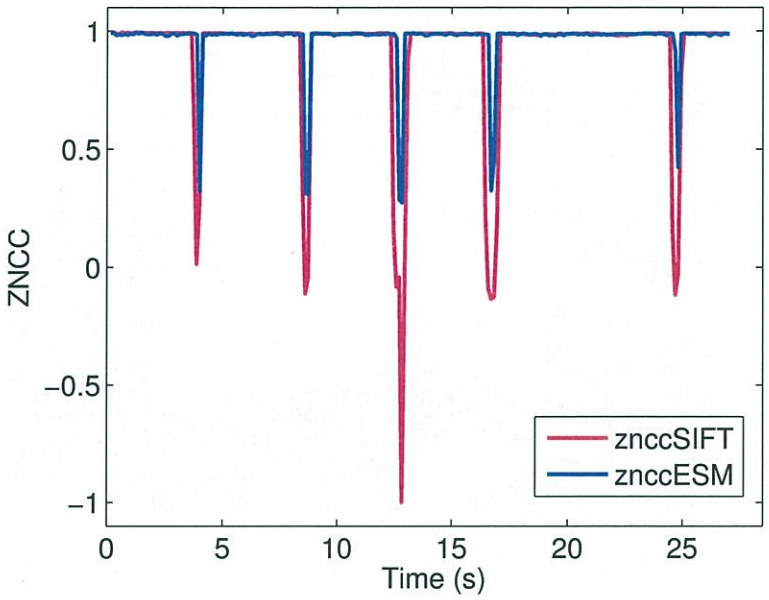


Figure 3.12: ZNCC Respecting to Time: when occlusion happens at $t = 3.61s$ and $t = 6.44s$, the ZNCC value falls down rapidly.

($t = 3.73$s after the first occlusion, $t = 6.58$s after the second occlusion in Figure 3.15). This has verified the robustness of our combination strategy to the occlusions.

3.4.4 Experiment III: Homography-based Visual Servo

The homography-based visual servo scheme is carried out on a real robot (*EPSON Pro Six PS5*). A reference image of the object is taken at the reference pose. Then the object is moved away from its reference pose (the black box at $t = 0.0$s in Figure 3.17). With this homography-based visual servo scheme, the robot is controlled to move the camera so as to keep the same relative position to the object. The image at $t = 1.99$s shows that finally the camera is correctly placed to its desired pose.

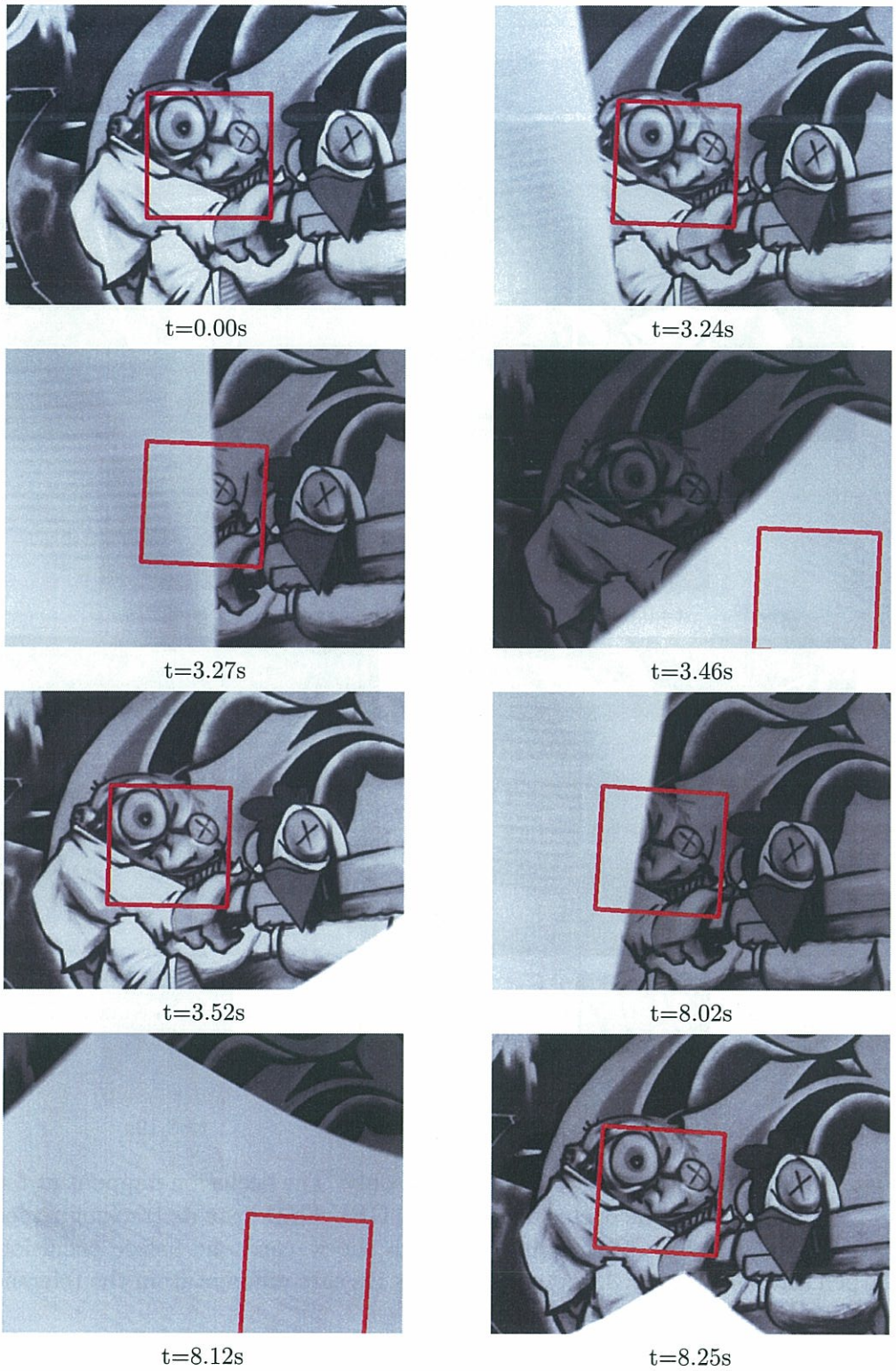


Figure 3.13: A sample sequence of input images is taken from the GPU thread. The red rectangular in the image denotes for the tracked region of GPU-ESM tracking algorithm.

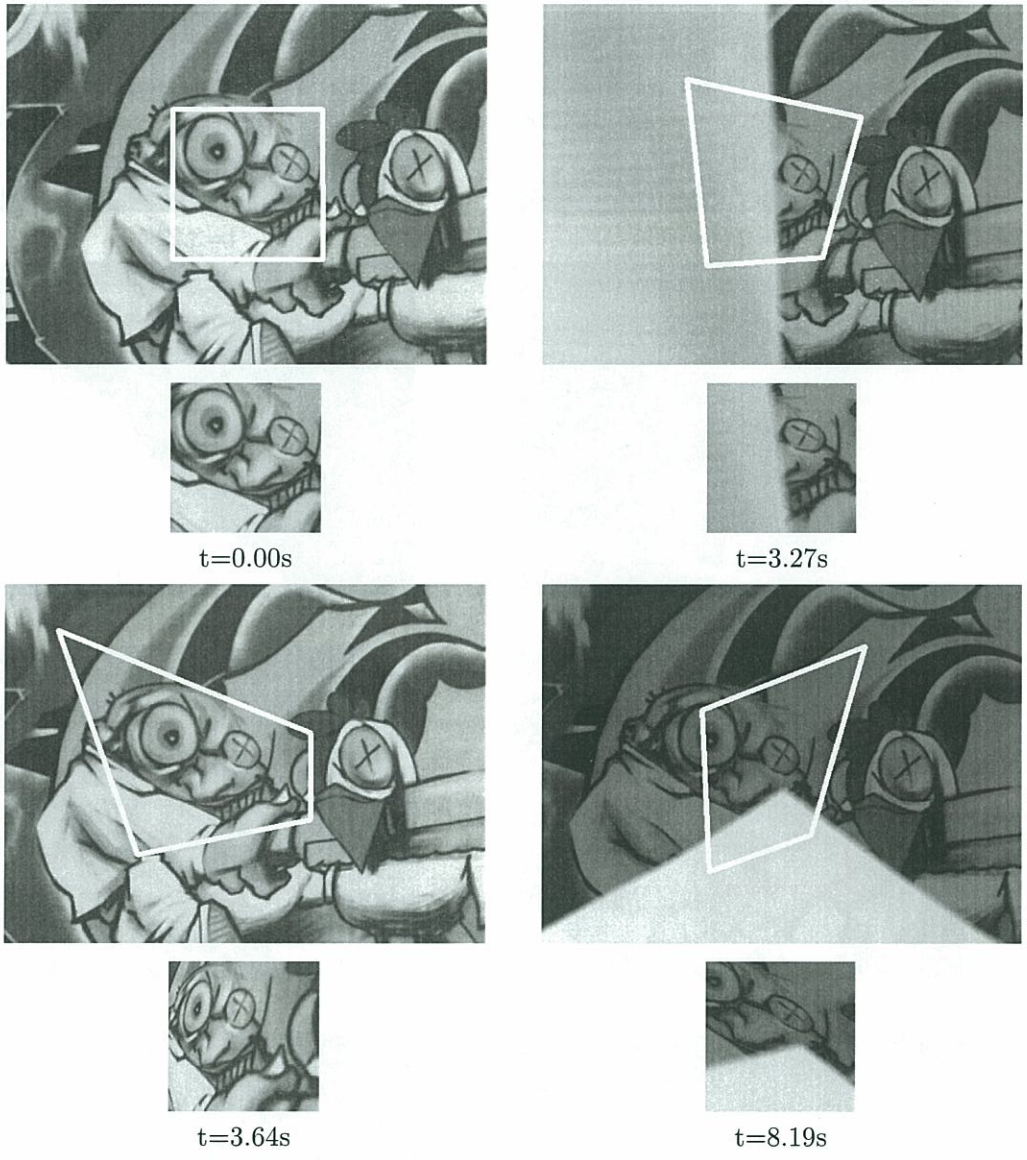


Figure 3.14: Occlusion test with GPU-ESM only. The occlusion happens at $t = 3.27$s and $t = 8.02$s. After the occlusion is moved, GPU-ESM losts its tracking region. The warped image at $t = 3.64$s is different from the warped one before occlusion. The warped images after $t = 3.52$s (such as $t = 8.19$s) are different from the reference one, which means the visual tracking fails.

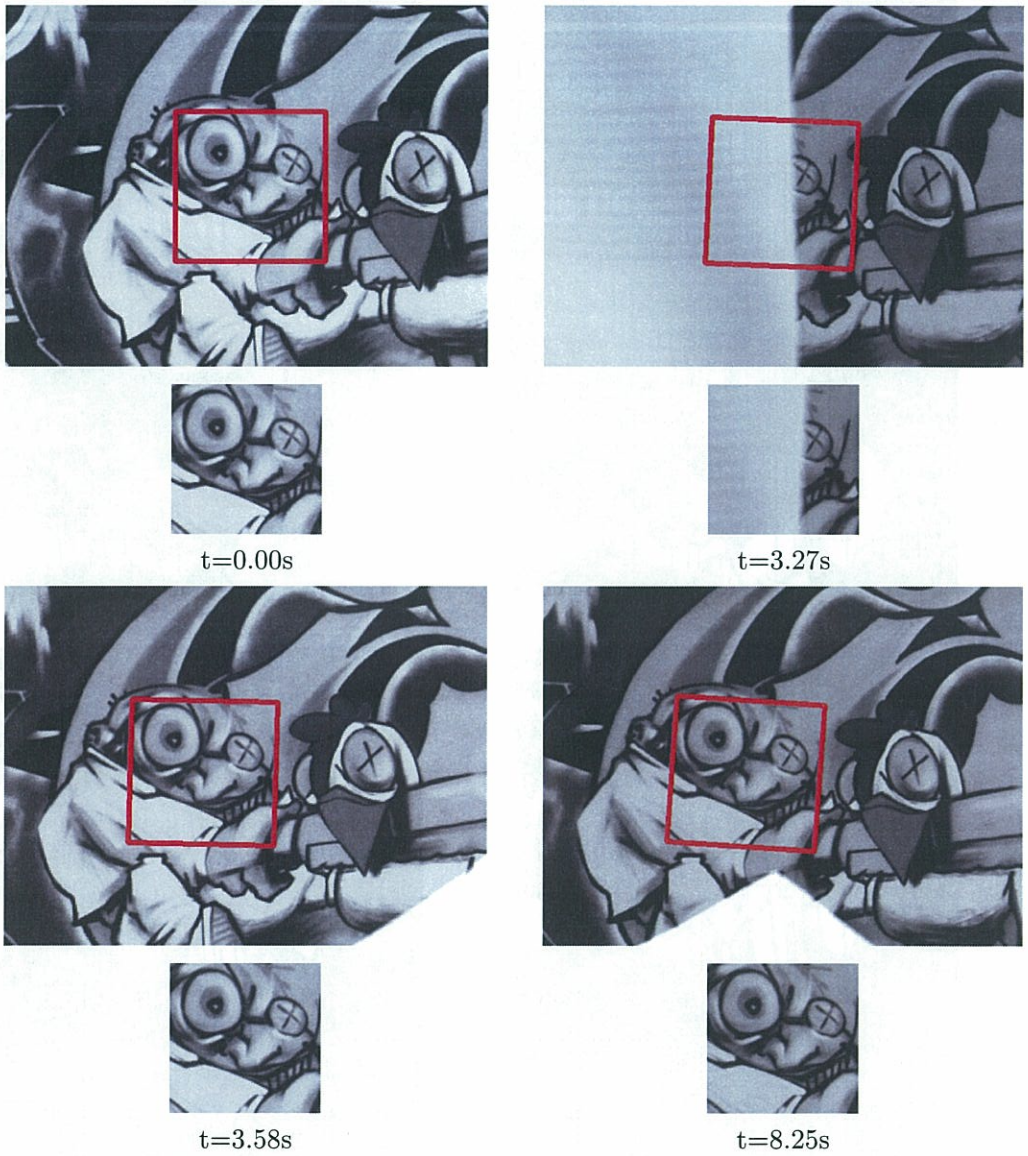


Figure 3.15: Evaluation of the combination strategy of GPU-ESM and GPU-SIFT with occlusion cases. The occlusion happens at $t = 3.27$s and $t = 8.02$s. After the occlusion is moved, GPU-ESM losts its tracking in Figure 3.14. Meanwhile our combination method can continue tracking ($t = 3.58$s, $t = 8.25$s). The corresponding ZNCC values are shown in Figure 3.12. This comparison shows that our combination strategy is effective to deal with occlusion cases.

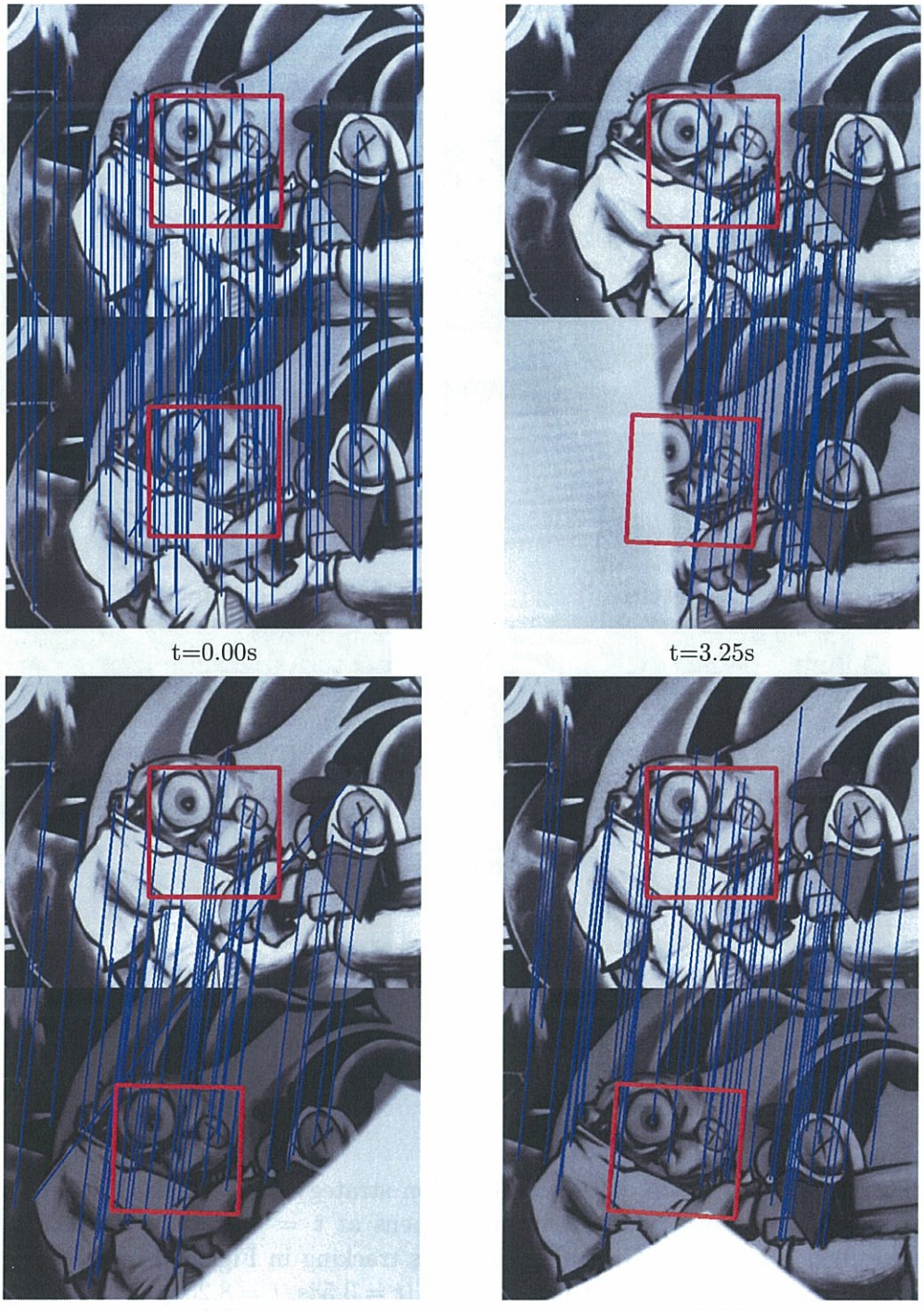


Figure 3.16: SIFT matching. After the occlusion is moved ($t = 3.52$s, $t = 8.21$s), homography from the SIFT matching method is loaded into GPU-ESM tracking algorithm so that it can continue tracking the region.

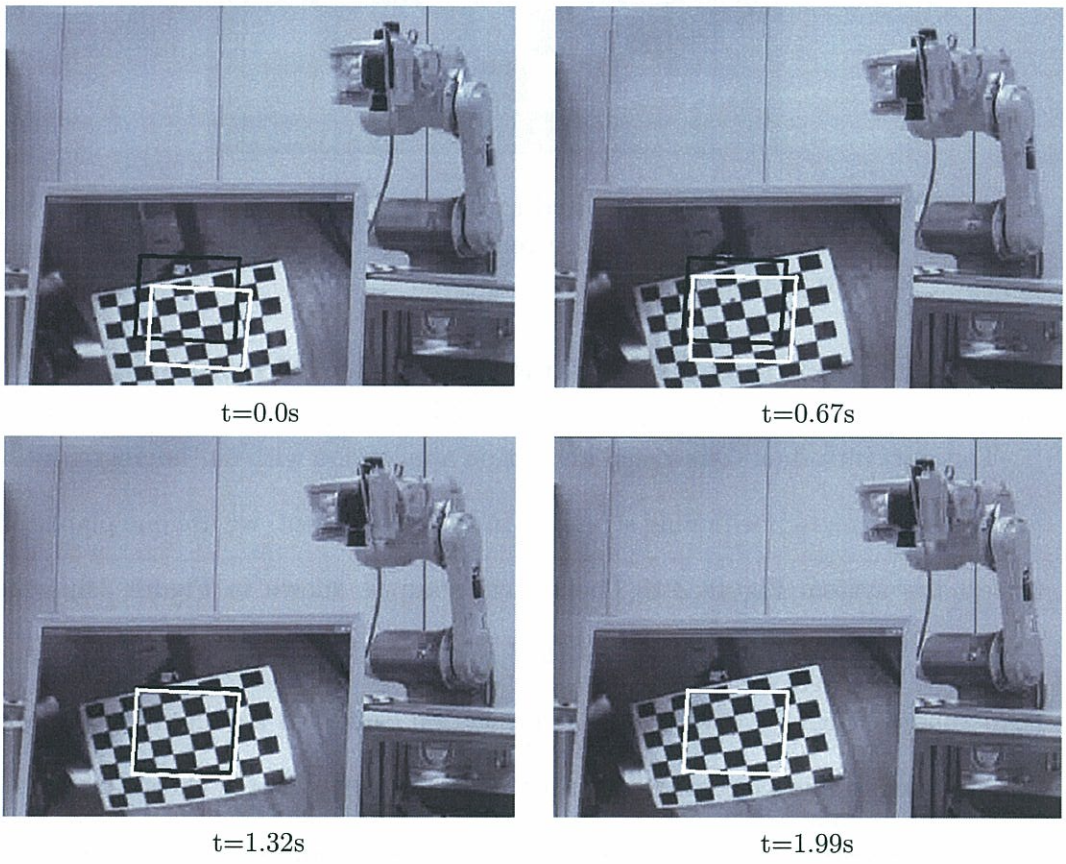


Figure 3.17: Homography-based Visual Servo. The black and white boxes describe the reference and current positions of the tracking area separately. The image at $t = 1.99$s shows that the visual servo scheme can work well (both boxes coincide finally).

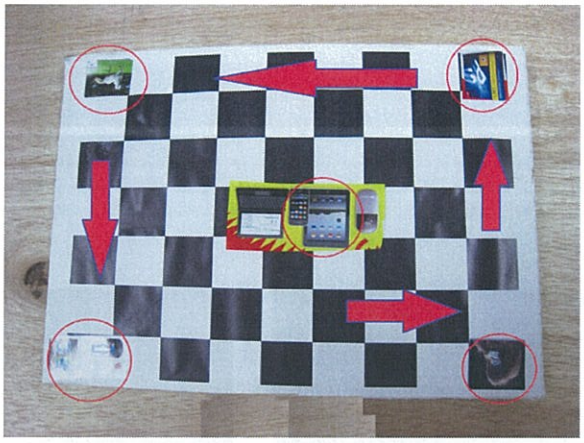
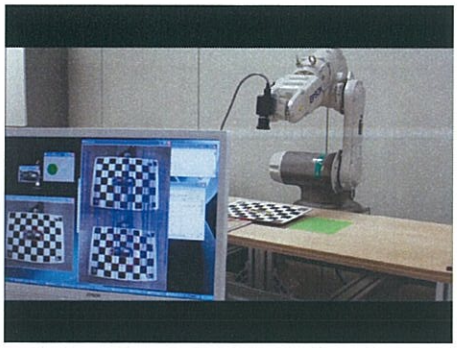


Figure 3.18: One planar object is used in the visual inspection application. Several inspection regions are marked by the red circles. The arrows show the inspection sequence of the regions.

3.4.5 Experiment IV: Visual Inspection Application

This part introduces one visual inspection application with our homography-based visual servo camera positioning strategy. In this experiment, we use one planar object to test the system Figure 3.18.The system setup is shown in Figure 3.4, which is composed of a six DOF robot (*EPSON Pro Six PS5*) and an IEEE 1394 camera (*Grasshopper GRAS-03K2M/C*). The inspection camera is mounted on the robot.

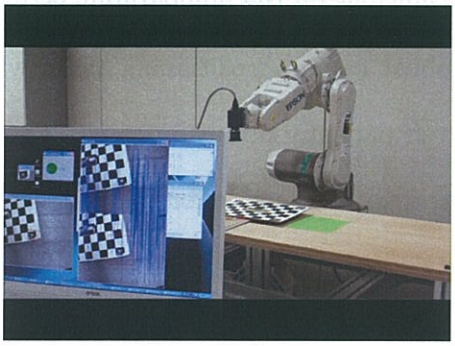
An image sequence extracted from the experiment video is shown in 3.19. The whole set of images for the training process is shown from (a) to (d). Then when the planar object is put at random initial pose, the combination strategy estimates the homography and uses it with homography-based visual servo to move the camera to new poses. The set of images of the inspection process with random initial pose is shown from (e) to (h). From this experiment, the efficiency of our camera positioning in the visual inspection system has been evaluated.



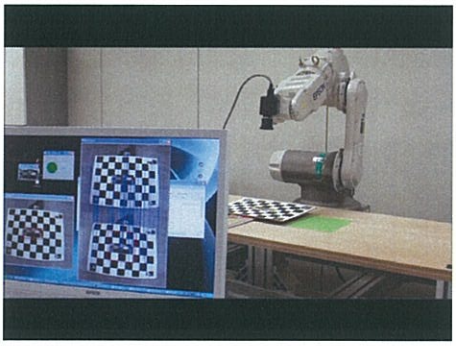
(a) Training pose 1

(b) Training pose 2

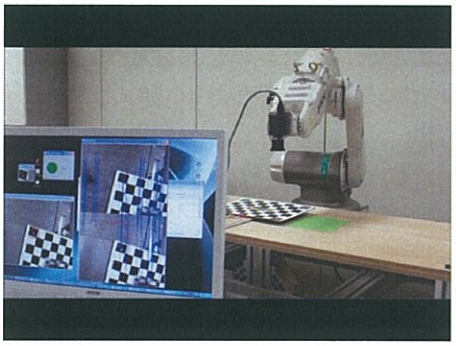
(c) Training pose 3



(d) Training pose 4

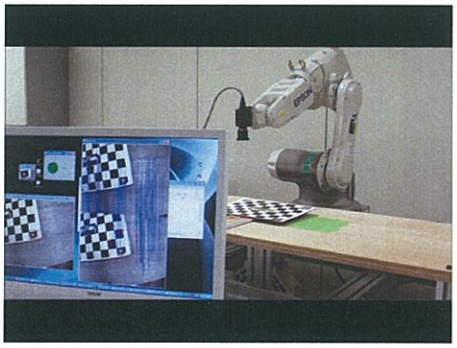


(e) New pose 1



(f) New pose 2

(g) New pose 3



(h) New pose 4

Figure 3.19: The whole set of images for the training process is shown from (a) to (d), while the set of images of the inspection process with random initial pose is shown from (e) to (h).

3.5 Conclusions

In this chapter, we have adopted a homography-based visual servo scheme to place a camera to its desired pose relative to a planar object. With the efficient combination of the GPU-ESM and GPU-SIFT algorithms, we have improved the system performances of processing speed, convergence range and the robustness to occlusions on the homography estimation. Therefore, our system can realize fast and robust camera positioning tasks within a large range of initial pose differences, or even partially occlusion cases. The experimental results show that our combination strategy is effective for this kind of camera positioning applications.

Chapter 4

Camera Positioning with an Estimated Pose

This chapter concentrates on solving the camera positioning task with respect to a 3D object. First the object pose relative to the camera is estimated with CAD model based methods. Then the camera is moved to its desired pose with the estimated pose information. In this chapter we have proposed a coarse-to-fine strategy to estimate the object's pose relative to the camera. In this strategy, the CAD model of the object is used in OpenGL to render computer graphics (CG) images. Then a global search for the maximum similarity between the real image and the CG images is carried out to find a pose solution. With this combination strategy, the system can provide fast and accurate pose estimation from a large initial differences range. This chapter describes the combination strategy in detail. The implementation of the combination strategy is introduced. Experiments results are shown to evaluate the efficiency of these algorithms.

4.1 Introduction

4.1.1 Camera Positioning with 3D Objects

In chapter 3 we have presented our work on placing a camera relative to a planar object. A more general application case is to place a camera with respect to 3D objects, so that the system can realize visual inspection, pick-and-place, and so on. Various camera positioning approaches have been realized in literature. One kind of the approaches are based on the relative pose information between the camera and the object. A survey of the pose estimation in detail has been shown in section 1.2.3. In this chapter our study focuses on pose estimation from 3D-2D correspondence (*model-to-image registration*). The CAD model of a 3D object is used to render computer graphics (CG) images. These pose estimation approaches are mainly classified into feature-based, local descriptor based, view(appearance)-based, and hybrid approaches.

Feature-based methods extract features such as edges, corner points from the CG images and real image respectively. By minimizing the feature differences between the real image and the projected CG images, the camera pose can be estimated with local optimization methods. A typical method is the edge-based pose estimation method by Drummond and Cipolla [94]. However, due to its local optimization nature, it needs a good guess of the initial pose, which is often impossible in real applications as the object might be put at random initial poses. Besides, extra work for specifying features in the CAD model is usually needed, such as defining the edge information in a CAD model. This work will be cumbersome for various objects as this work has been done for each kind of objects respectively.

Local descriptor-based methods such as SIFT matching perform poorly with those texture-less objects, e.g., a metal part or plastic part which is popular in industrial applications. Lacking of texture makes it difficult for the feature detection for further process.

View based methods estimate the pose by using local optimization method to reduce the image difference between the CG image and real image. They can provide accurate pose estimation when the initial difference is small, however, their local optimization nature causes them to perform poorly with large initial pose differences.

In above applications, the key problem is to find a global pose solution of the object relative to the camera. If the global pose has been estimated, local optimization methods can be used to refine the pose.

To deal with the local optima problem, we have proposed a global search with the view-based approach to find the pose estimation. In the view-based approach, an exhaust search is necessary to provide global pose estimation without prior knowledge of the object pose so that it can avoid the local optima problem. First a computer graphics (CG) image is rendered from each possible view/pose in the search space, then a comparison between the real image and the CG images is carried out to find one CG image with the maximum similarity with the real image. The pose to render this CG image is treated as the pose estimation required.

However, due to the huge search space, the running time of the whole "render-compare" loops for all the poses is a great challenge. Taking the example of a search in a 3DOF space of (20cm $\times$ 20cm $\times$ 360°) with resolution (1mm, 1mm, 1°) respectively, the number of poses is $200 \times 200 \times 360 = 14400000$. One "render-compare" loop for VGA size image takes about 7.5 ms, then the total time is about 108000 s (30 hours).

For a search in a 6DOF space, the search takes even more time, which makes it not applicable to the real applications.

To accelerate the search, we have proposed a lookup table method. A series of image features are calculated from the image moments of all the CG images and used in the lookup table to distinguish the CG images and find the one with maximum similarity with the real image. Experiments show that using more kinds of image features can make them more distinguishable, however, more kinds of features means larger dimension of the lookup table which requires more memory space. Therefore we adopt the principal component analysis (PCA) on these features to reduce the table dimension. With PCA analysis the features' distribution has been improved and the memory space for the lookup table has been effectively reduced while the system is still as distinguishable as before PCA processing. By searching the maximum similarity in a smaller space with the lookup table, fast and reliable pose estimation has been obtained.

After the global search for the initial pose solution, the pose estimation is refined by local optimization approaches to provide a more accurate pose to move the camera. Besides the pose estimation, another problem lies in the difference between the virtual camera and a real camera, due to the calibration errors in camera parameters $\boldsymbol{K}$ and the coordinate transformation from the object frame to the robot base frame. To accurately place the camera to its desired pose, we have adopted visual servo technique in our system.

4.1.2 Goals and Contributions

In this chapter our goal is to realize the camera positioning with respect to 3Dobjects. Our contributions are mainly as follows.

we propose a coarse-to-fine strategy for pose estimation so as to move the camera with respect to 3D objects. Compared with other methods, the global search in the coarse step for the maximum similarity between the real image and those CG image has ensured reliable initial pose estimation within a large initial pose range meanwhile the local optima as in other local methods have been effectively avoided. As there is no feature detection process in our strategy, feature specification work in the features-based methods is omitted. With the combination strategy, a fine step of local minimization approaches ensures high accuracy pose estimation for moving a camera. Finally, the visual servo technique has effectively reduced the positioning error and accurately moved the camera to its desired pose.

Another contribution lies in the acceleration of the global search. To accelerate the search, we adopt several optimization approaches, including resizing the image for similarity comparison and using a lookup table strategy. We adopt the PCA processing to reduce the dimension of the lookup table by which more image features from the image moments can be used to make the images more distinguishable. By searching the maximum similarity in a much smaller space in the lookup table with more features, a remarkable speedup of the pose estimation has been obtained.

The rest is organized as follows. Section 4.2 describes the related algorithms adopted in this chapter. Section 4.3 introduces the implementation of these algorithms in detail. Section 4.4 describes the experimental results to evaluate the performances. Section 4.5 concludes this chapter.

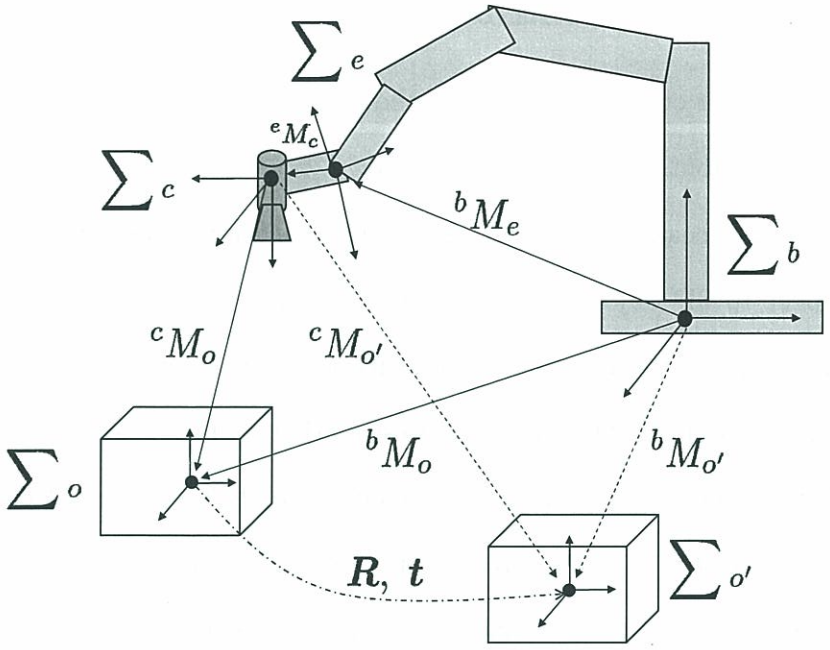


Figure 4.1: Coordinates frames in a robot system. The frames $\sum_o$, $\sum_b$, $\sum_c$, and $\sum_e$ are assigned in the object, robot base, camera and end effector respectively.

4.2 Related Algorithms

4.2.1 Camera Positioning with an Estimated Pose

The robot system configuration is shown in Figure 4.1. A camera is attached on a robot end effector. Four coordinate frames are mentioned in this framework: the camera frame $\sum_c$, the end-effector frame $\sum_e$, the robot base frame $\sum_b$, and the object frame $\sum_o$. The relation between these frames are expressed as ${}^bM_o$, ${}^cM_o$, ${}^bM_e$ respectively as in Figure 4.1.

The goal of the camera positioning task is to move the camera to keep the same relative pose in the object frame when the object moves. Let ${}^bM_o$ and ${}^bM_{o'}$ be the object pose in the robot base frame before and after a translation $\boldsymbol{t}$ and a rotation $\boldsymbol{R}$, meanwhile the new end-effect pose ${}^bM_{e'}$ is required to move the robot.

When the object motion $[\boldsymbol{R}, \boldsymbol{t}]$ is defined relative to its own coordinate system as $[\boldsymbol{R}_o, \boldsymbol{t}_o]$, i.e., translation along its x, y and z axis and rotation about its x, y and z axis, the postmultiplication[1] operation is used to update the object pose relative to the camera frame (${}^cM_o$) and robot base frame (${}^bM_o$) respectively as

$$ {}^cM_{o'} = {}^cM_o \begin{bmatrix} \boldsymbol{R}_o & \boldsymbol{t}_o \\ \mathbf{0} & 1 \end{bmatrix}, \tag{4.1} $$

$$ {}^bM_{o'} = {}^bM_o \begin{bmatrix} \boldsymbol{R}_o & \boldsymbol{t}_o \\ \mathbf{0} & 1 \end{bmatrix}. \tag{4.2} $$

Meanwhile, when the same object motion $[\boldsymbol{R}, \boldsymbol{t}]$ is expressed in the robot base frame as $[\boldsymbol{R}_b, \boldsymbol{t}_b]$, the object pose relative to the robot base frame (${}^bM_o$) can be updated by a premultiplication operation as

$$ {}^bM_{o'} = \begin{bmatrix} \boldsymbol{R}_b & \boldsymbol{t}_b \\ \mathbf{0} & 1 \end{bmatrix} {}^bM_o. \tag{4.3} $$

To keep the camera with a same relative pose to the object, the camera/end-effector should have the same motion as the object. Therefore, the update of the end-effector pose relative to the robot base frame (${}^bM_e$) is in the same way of a premultiplication operation

$$ {}^bM_{e'} = \begin{bmatrix} \boldsymbol{R}_b & \boldsymbol{t}_b \\ \mathbf{0} & 1 \end{bmatrix} {}^bM_e. \tag{4.4} $$

Suppose the $[\boldsymbol{R}_o, \boldsymbol{t}_o]$ in (4.1) can be obtained from the CAD model matching

[1]The relationship of postmultiplication and premultiplication is discussed in detail in section B.2.

approaches, the desired new pose ${}^{b}M_{e'}$ in the robot motion can be calculated from (4.2), (4.3), (4.4) as

$$ {}^{b}M_{e'} = {}^{b}M_{o} \begin{bmatrix} \boldsymbol{R}_o & \boldsymbol{t}_o \\ \mathbf{0} & 1 \end{bmatrix} {}^{b}M_{o}^{-1} \, {}^{b}M_{e}, \tag{4.5} $$

where ${}^{b}M_{e}$ can be calculated from the robot joint angles with D-H convention, ${}^{b}M_{o}$ can be manually calibrated in a calibration step. Therefore, the robot can move the camera to its desired pose with the ${}^{b}M_{e'}$ information. Hence the problem consists in finding the object transformation $[\boldsymbol{R}_o, \boldsymbol{t}_o]$ in the camera frame.

4.2.2 Pose Estimation with a Lookup Table

This part mentions the pose estimation with a lookup table in the coarse step. The image features are calculated from the spatial and central moments of all the CG images. The spatial moment $m_{p,q}$ of order (p, q) is defined as

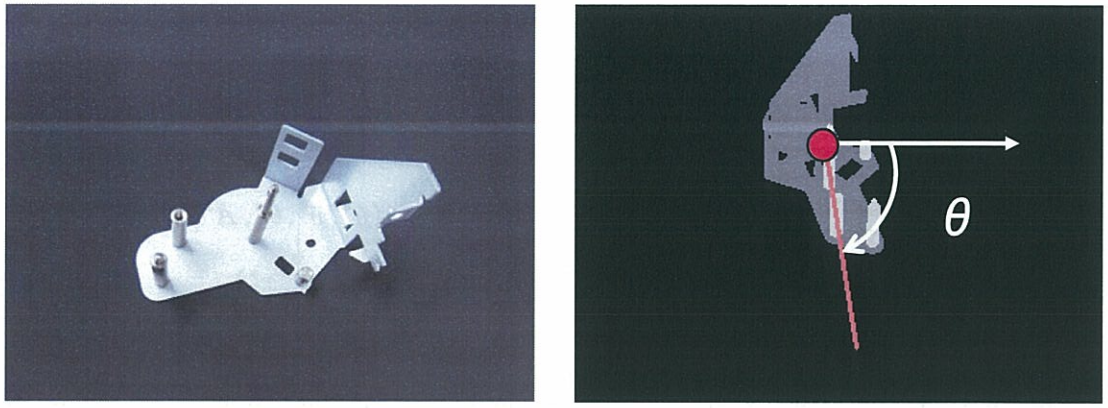
$$ m_{p,q} = \sum_{u,v} u^p v^q I(u, v), \tag{4.6} $$

where the summation is performed for all pixels in the image. The central moment $\mu_{p,q}$ of order (p, q) is defined as

$$ \mu_{p,q} = \sum_{u,v} (u - \bar{u})^p (v - \bar{v})^q I(u, v), \tag{4.7} $$

where $(\bar{u}, \bar{v})$ is the center of gravity of the object, which can be calculated by

$$ \bar{u} = \frac{m_{1,0}}{m_{0,0}}, \ \bar{v} = \frac{m_{0,1}}{m_{0,0}}. \tag{4.8} $$



(a) A gear base object (b) One CG image

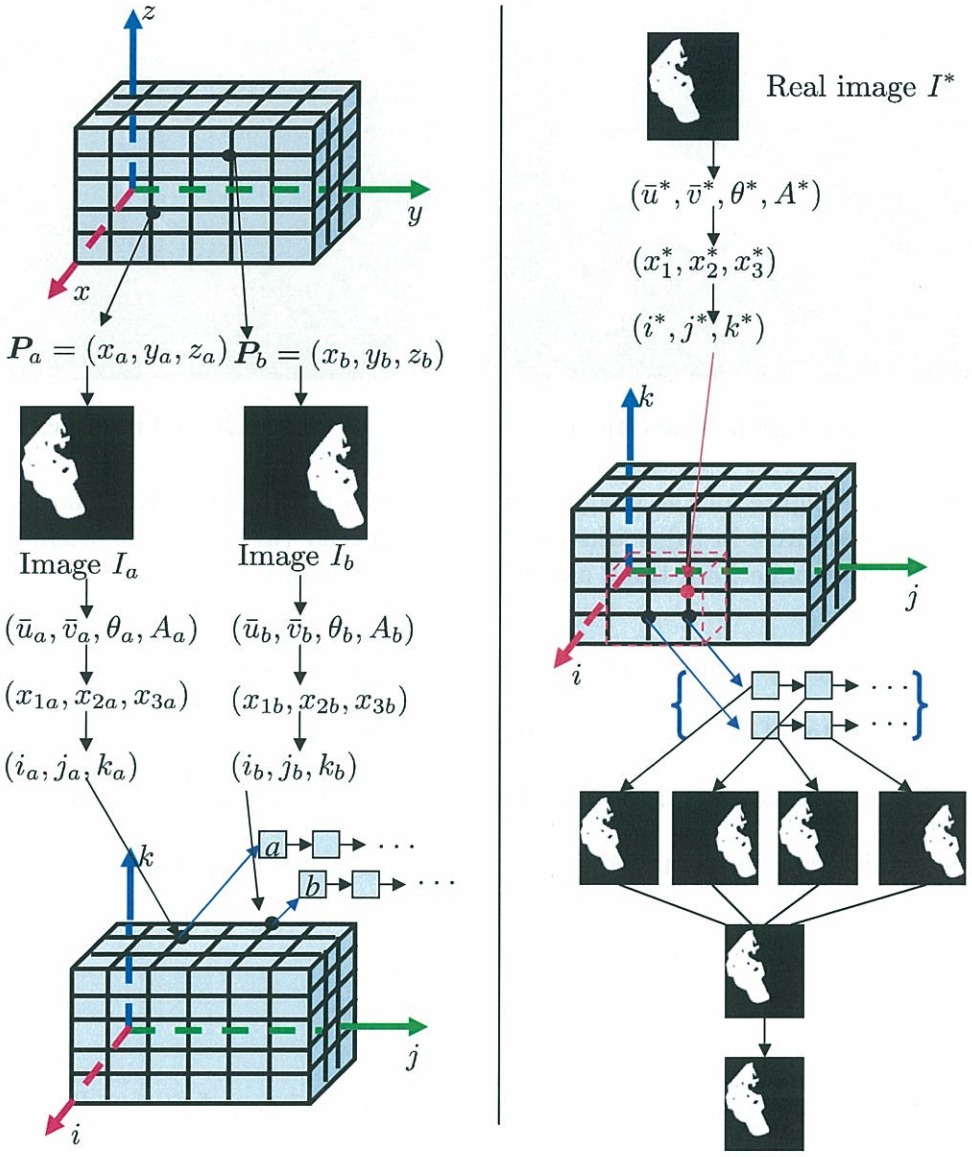
Figure 4.2: A figure for the image features in a CG image. (a) is the image of a gear base. The red circle in (b) marks the center of gravity $(\bar{u}, \bar{v})$ and the orientation θ is defined as the angle from the u axis to the main axis of the object.

rom the image moments several image features can be calculated, such as the area $A = m_{0,0}$, center of gravity $(\bar{u}, \bar{v})$ of the object, the orientation θ of the object (Figure 4.2). The orientation θ is defined as the angle from the u axis to the main axis of the object, which can be calculated from second-order central moments:

$$\tan(2\theta) = \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}. \tag{4.9}$$

In our applications, we adopt the $(\bar{u}, \bar{v}, \theta, A)$ as the image features. To reduce the dimension of the lookup table, we use the principal component analysis (PCA) on these features. The process flowchart is shown in Figure 4.4.

The whole pose estimation strategy has been divided into two phases, the *offline learning* phase for establishing a lookup table and the *online search* phase for searching a pose in the lookup table (an example is shown in Figure 4.3).



(a) Offline Learning

(b) Online Search

Figure 4.3: An example of global search with a lookup table for estimation of 3 DOF x, y, z of the pose (suppose orientation keeps the same). In the offline liearning process, two CG images I_a and I_b are rendered from two pose $\boldsymbol{P}_a$ and $\boldsymbol{P}_b$. Then features $(\bar{u}, \bar{v}, \theta, A)$ are calculated and projected to (x_1, x_2, x_3) with PCA projection. A 3-element integer index (i, j, k) is calculated from the features (x_1, x_2, x_3) and the pose index a and b are filled in the 3D lookup table. In the online search process, the features are extracted from the real image I^* as the same way. Then a cell around the index (i^*, j^*, k^*) are chosen and a group of candidate poses are extracted. Then for each candidate pose, a CG image is rendered and compared with the real image. The pose to produce the CG image with maximum similarity as the real image is treated as the poes estimation.

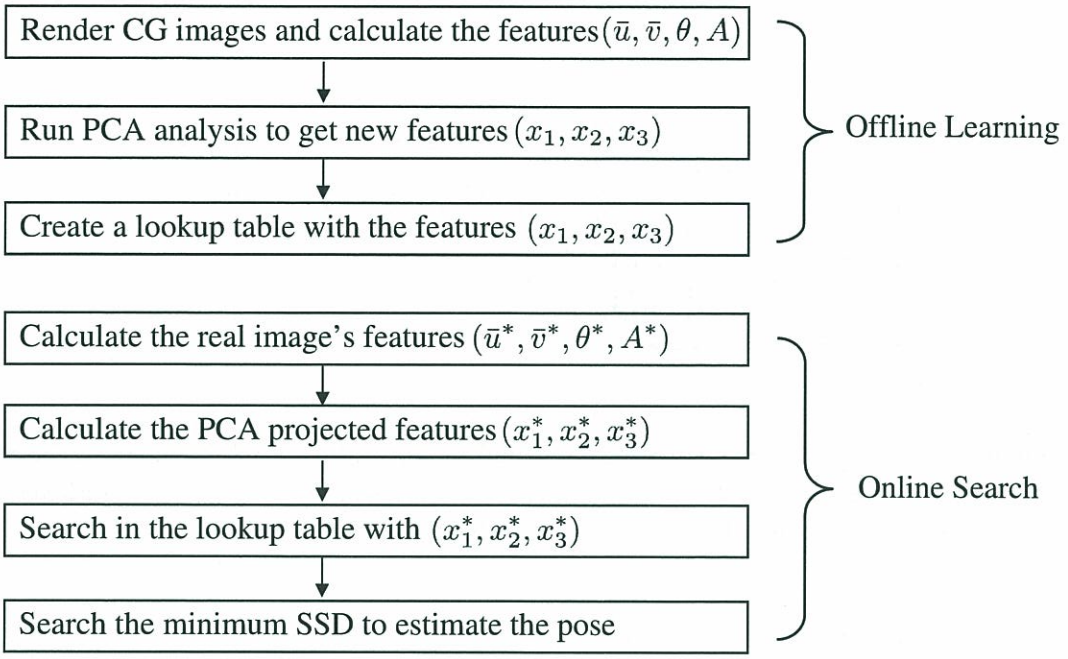


Figure 4.4: The process flowchart of pose estimation with alookup table. The whole process is divided into two phases, the *offline learning* phase for establishing a lookup table and the *online search* phase for pose estimation with the lookup table.

Offline Learning. This phase creates a lookup table in offline. The running time for this phase depends on the size of the search space and the resolution of the search. This kind of offline learning is only done once.

- Set the search space size (the data range of each degree to be estimated) and render a CG image for each pose in the search space and calculate the features $(\bar{u}, \bar{v}, \theta, A)$ of each CG image and save the features into files.

- Run the PCA analysis on the features to reduce the dimension of the lookup table. The features $(\bar{u}, \bar{v}, \theta, A)$ are projected to three new elements $(\boldsymbol{X} = (x_1, x_2, x_3))$.

- Calculate the range $range_n$ of the feature x_n $(n = 1, 2, 3)$ by searching the minimum and maximum values $x_{n\,min}$ and $x_{n\,max}$ in each dimension with $range_n =$

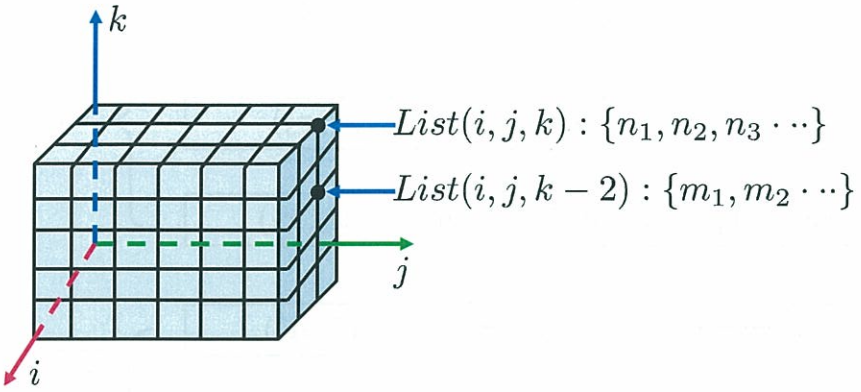


Figure 4.5: A figure of the three dimensional lookup table. Each element in the table is a pointer pointing to a list with index (i, j, k). Each list contains the indexes of a set of poses, from which the CG images have similar features.

$x_{n\,max} - x_{n\,min}$, $(n = 1, 2, 3)$.

With the new features $\boldsymbol{X}$, a three dimensional lookup table is created (Figure 4.5). As the length of an array and the index of an element have to be integer, a round operation to convert float variable *data* to integer is needed (expressed as round(*data*)).

- The size of each dimension of the table is $l_n = \text{round}(range_n)$, $n = 1, 2, 3..$
- Each element in the table is a pointer pointing to a list with index (i, j, k). The range of (i, j, k) is $[0, l_1 - 1]$, $[0, l_2 - 1]$ and $[0, l_3 - 1]$ respectively.
- For feature $\boldsymbol{X}_a = (x_{1a}, x_{2a}, x_{3a})$ of the CG image from pose index a, calculate a 3-element index by

$$\begin{aligned} i_a &= \text{round}(x_{1a} - x_{1min}), \\ j_n &= \text{round}(x_{2a} - x_{2min}), \\ k_n &= \text{round}(x_{3a} - x_{3min}). \end{aligned} \tag{4.10}$$

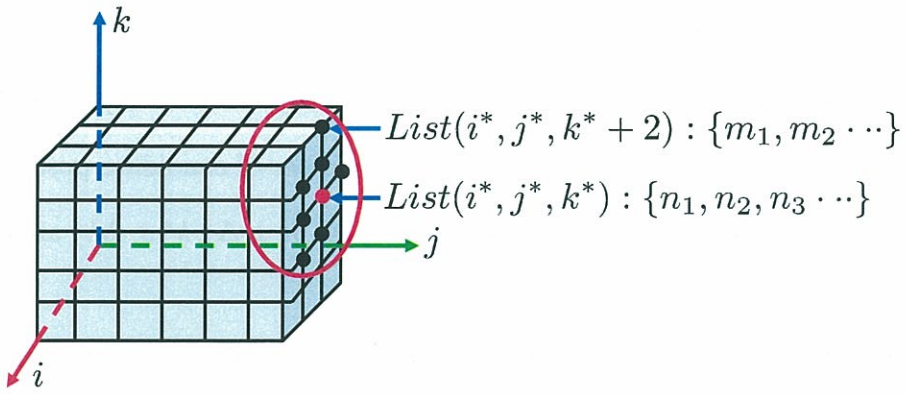


Figure 4.6: A figure for searching in the lookup table. After calculate the new 3-element index (i^*, j^*, k^*) (red dot in the figure) from the real image, all the pose indexes in the neighbor lists (e.g., black dots in the red circle) $(i^* \pm \Delta i, j^* \pm \Delta j, k^* \pm \Delta k)$ are loaded. Then the "render-compare" strategy is used with this small number of poses and fast pose estimation can be obtained.

- Put the pose index a into the list whose index is (i_a, j_a, k_a). This operation is repeated for all the poses. Until now a lookup table has been established with the PCA projected features. The CG images rendered from the poses in one list have similar features $(\bar{u}, \bar{v}, \theta, A)$.

Online Search. After capturing a real image from a camera, the online search for pose estimation is realized as follows (Figure 4.6).

- Calculate the features $(\bar{u}^*, \bar{v}^*, \theta^*, A^*)$ of the real image.

- Project these features with the same PCA projection matrix as in the offline learning to get three new features $\boldsymbol{X}^* = (x_1^*, x_2^*, x_3^*)$. Calculate a 3-element index from the features by

$$\begin{aligned} i^* &= \text{round}(x_1^* - x_{1min}), \\ j^* &= \text{round}(x_2^* - x_{2min}), \\ k^* &= \text{round}(x_3^* - x_{3min}), \end{aligned} \tag{4.11}$$

- Choose a cell around the calculated index with cell size of $(\Delta i, \Delta j, \Delta k)$ in each dimension, where $(\Delta i, \Delta j, \Delta k)$ is automatically set from experimental experience respectively. From the cell a small number of candidate poses in the lists whose indexes are within the cell of $(i^* \pm \Delta i, j^* \pm \Delta j, k^* \pm \Delta k)$ are extracted.

- Run the "render-compare" processing on each candidate pose and calculate the similarity between the CG image from this pose and the real image.

- Search for the maximum similarity value within this small set and the pose with maximum similarity is treated as the pose estimation. By this means, the pose estimation has been realized.

4.2.3 Similarity Measure

After finding a small number of candidate CG images by searching in the lookup table, a robust and effective similarity measure is crucial for the pose estimation. Several similarity measures have been proposed in literature, such as the Sum of Squared Differences (SSD), Zero-mean Normalized Cross Correlation (ZNCC), Mutual Information (MI) [138]. A comparison of these measures can be found in [139]. In our evaluation experiments, they have similar performances in our applications. Therefore in this framework, we adopt the fastest one –SSD– as a measure of the similarity between a CG image and a real image.

When the two images are same, the SSD value has a minimum value 0. By this means, a best match of a CG image and a real image can be realized by searching the minimum SSD value.

4.2.4 Local Optimization to Refine the Pose

In this part we describe the local minimization techniques to refine the pose estimation in the fine step. A Gauss-Newton method is adopted to find an optimal pose by minimizing the difference between the CG image and the real image. For particular cases where the edges can be extracted from the image, an edge-based optimization method has been adopted for optimal pose estimation.

(a) Pose Estimation by CG Images Matching

In our system, the geometry information of the object CAD model is used in OpenGL to render a series of CG images from different virtual camera poses. By minimizing the intensity difference between the current image and the CG image, an optimal pose solution is estimated.

With the standard pin-hole camera model[2] for the perspective projection, given a known object CAD model and the camera intrinsic parameters $\boldsymbol{K}$ and extrinsic matrix $\boldsymbol{P}$, a CG image I can be rendered from OpenGL. This rending process is like taking one photo of the object from a virtual camera with parameters of $\boldsymbol{K}$ and $\boldsymbol{P}$. Here $\boldsymbol{P}$ is the object pose consisting of the orientation $\boldsymbol{R}$ and position $\boldsymbol{t}$ of the object.

When the intrinsic $\boldsymbol{K}$ is fixed, the CG image I can be expressed as $I(\boldsymbol{P})$ as it changes with the extrinsic parameters $\boldsymbol{P}$. A current image I^* is captured from a real camera with an unknown camera pose $\boldsymbol{P}^*$. Therefore, the camera pose estimation consists in finding the parameter $\boldsymbol{P}$ which minimizes the image difference between the CG image $I(\boldsymbol{P})$ from a virtual camera and the current image I^*

$$d = I(\boldsymbol{P}) - I^*(\boldsymbol{P}^*). \tag{4.12}$$

[2]The pin-hole camera model is described in Appendix A on page 155 in detail.

In general, this problem is solved by a sum-of-squared difference minimization, which consists in minimizing the following function

$$f(\boldsymbol{P}) = \frac{1}{2}(I(\boldsymbol{P}) - I^*(\boldsymbol{P}^*))^{\mathrm{T}}(I(\boldsymbol{P}) - I^*(\boldsymbol{P}^*)). \tag{4.13}$$

Linearize the function at $\boldsymbol{P}^*$ with the second-order Taylor series approximation,

$$f(\boldsymbol{P}) = f(\boldsymbol{P}^*) + \nabla f(\boldsymbol{P})^{\mathrm{T}}\Delta\boldsymbol{P} + \frac{1}{2}\Delta\boldsymbol{P}^{\mathrm{T}}\nabla^2 f(\boldsymbol{P})\Delta\boldsymbol{P}, \tag{4.14}$$

where $\Delta\boldsymbol{P} = \boldsymbol{P} - \boldsymbol{P}^*$. We define the image Jacobian matrix as

$$\boldsymbol{J}(\boldsymbol{P}) = \frac{\partial I}{\partial \boldsymbol{P}}. \tag{4.15}$$

With the Gauss-Newton method, we can replace the items in (4.14) as

$$\nabla f(\boldsymbol{P})^{\mathrm{T}} = \boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}(I(\boldsymbol{P}) - I^*(\boldsymbol{P}^*)), \nabla^2 f(\boldsymbol{P}) = \boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{P}). \tag{4.16}$$

Thus (4.14) becomes

$$f(\boldsymbol{P}) = f(\boldsymbol{P}^*) + \boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}(I(\boldsymbol{P}) - I^*(\boldsymbol{P}^*))\Delta\boldsymbol{P} + \frac{1}{2}\Delta\boldsymbol{P}^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{P})\Delta\boldsymbol{P}. \tag{4.17}$$

Take the derivative with respect to $\boldsymbol{P}$ and set it to zero to find a solution

$$\boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}(I(\boldsymbol{P}) - I^*(\boldsymbol{P}^*)) + \boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{P})\Delta\boldsymbol{P} = 0. \tag{4.18}$$

Therefore, the update can be solved from (4.18) by adding a positive gain λ,

$$\Delta \boldsymbol{P} = -\lambda(\boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{P}))^{-1}\boldsymbol{J}(\boldsymbol{P})^{\mathrm{T}}(I(\boldsymbol{P}) - I^{*}(\boldsymbol{P}^{*})). \tag{4.19}$$

To compute the Jacobian matrix in the Gauss-Newton method, we adopt an approximation of the Jacobian matrix as:

$$\boldsymbol{J}(\boldsymbol{P}) = \frac{\partial I}{\partial \boldsymbol{P}} \simeq \frac{I(\boldsymbol{P} + \Delta \boldsymbol{P}) - I(\boldsymbol{P})}{\Delta \boldsymbol{P}}. \tag{4.20}$$

The small displacement of $\Delta \boldsymbol{P}$ in each of the six degrees of freedom is chosen manually from the practical experiments. The gain λ in (4.19) is manually set to 0.1. By updating the pose $\boldsymbol{P}$ iteratively with $\Delta \boldsymbol{P}$ from an initial pose $\boldsymbol{P}_0$, finally an optimal pose estimation can be obtained.

(b) Edge-based Pose Estimation

For particular cases where the edge can be extracted from the image, we adopt the edge-based pose estimation method by Drummond and Cipolla [94]. In this frame, the pose computation problem consists in estimating the interframe motion $\boldsymbol{M}_t$ between the current pose $\boldsymbol{P}_t$ and the previous pose $\boldsymbol{P}_{t-1}$,

$$\boldsymbol{P}_t = \boldsymbol{P}_{t-1}\boldsymbol{M}_t, \tag{4.21}$$

where $\boldsymbol{M}_t$ forms a 4×4 matrix representation of the rigid body motions in Lie group $SL(3)$. $\boldsymbol{P}_t$ and $\boldsymbol{P}_{t-1}$ represent the pose at time t and $t-1$. From the Lie algebra, $\boldsymbol{M}_t$

can be expressed in the exponential map of generators $\boldsymbol{G}_i$,

$$\boldsymbol{M}_t = \exp(\boldsymbol{\alpha}) = e^{\sum_{i=1}^{6} \alpha_i G_i} \tag{4.22}$$

where the generators $\boldsymbol{G}_i$ are generally taken as the translations in the x, y and z directions and rotations about the x, y and z axes in (4.23). The $\boldsymbol{\alpha} \in \mathbb{R}^6$ describes the six velocities corresponding to the six instantaneous displacements. Thus, the task of tracking becomes finding those α_i to describe the interframe motion.

$$\boldsymbol{G}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{G}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{G}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{G}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{G}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.23}$$

With Lie algebra and pin-hole camera model, a least-squares approach is used to fit the motion between adjacent frames. To increase the robustness to outliers, a robust M-estimator is substituted for the least-squares estimator.

4.3 Implementation

4.3.1 Implementation of the Coarse-to-fine Strategy

We have proposed a coarse-to-fine strategy to place a camera with respect to 3D objects. The process flowchart is shown in Figure 4.7.

- Initial pose estimation is obtained by view-based pose estimation method in the coarse step. The initial pose is estimated by a global search for the maximum similarity between the real image and those images rendered from Computer Graphics (CG image), meanwhile the local optima as in other methods have been avoided. The related algorithms have been described in section 4.2.2 in detail.

- As the pose estimation is limited by the resolution in each parameters in the offline learning step for making the lookup table, we use a smaller step size to search again after finding the initial pose solution with the lookup table. To do this, a small group of poses around the coarse estimation are calculated and the "render-compare" loop is run on all these poses. Therefore, a more accurate solution can be estimated by this "neighbor search".

- Then the pose estimation is refined by local optimization approaches in the fine step to provide a more accurate pose to move the camera. The related algorithms have been described in section 4.2.4.

- Finally, a visual servo scheme is adopted to reduce the error from system calibration and ensure the camera to be placed to its desired pose.

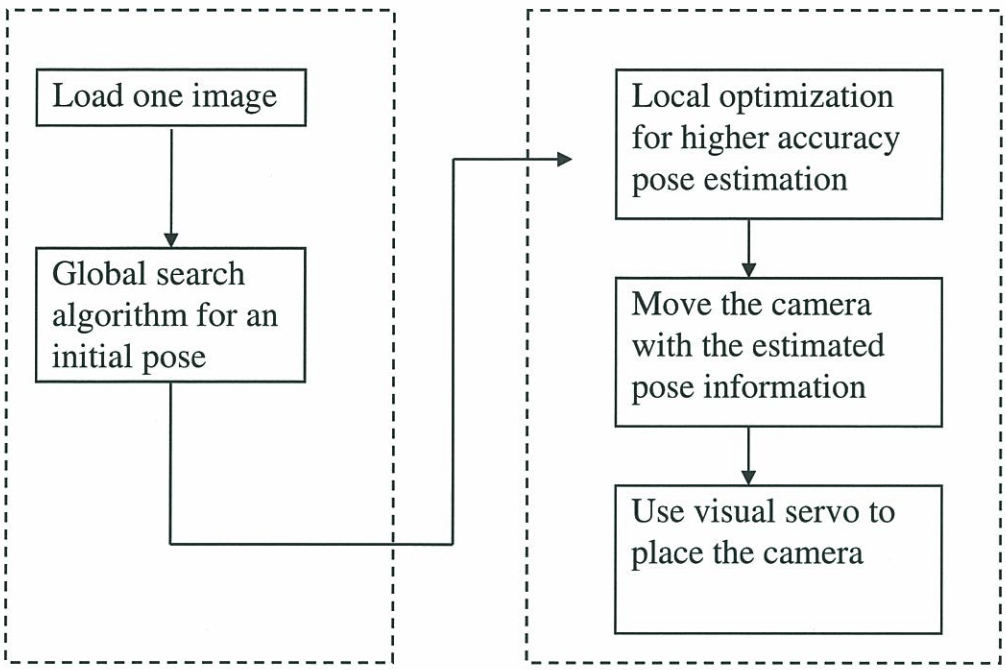


Figure 4.7: Process flowchart of a coarse-to-fine strategy adopted to locate a camera with respect to 3D objects. A global matching algorithm offers coarse initial pose estimation in the coarse step. In the fine step, the pose estimation is refined by local optimization approaches. With this pose the camera is moved to near its final pose. To reduce the error due to system calibrations, a visual servo scheme is adopted finally to locate the camera to its desired pose.

4.3.2 OpenGL Correspondence Analysis

For Computer Graphics, we choose OpenGL to render the CG images. As a cross-platform standard for 3D rending, it has a well-defined specification, including the geometric transformation pipeline.To simulate the real camera and object in OpenGL, it is necessary for correspondence analysis, i.e., how to assign the real camera's intrinsic parameters $\mathbf{K}$ and extrinsic parameter $\mathbf{P}$ (composed of rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$) to the OpenGL CG pipeline so that the rendered CG image is nearly the same as the one taken from a real camera (despite the lighting differences). We have adopted

a similar method as in [140]. The correspondence analysis is based on the comparison of pin-hole camera model and OpenGL CG pipeline.

The pin-hole camera model gives the relationship between 2D image pixels $[u, v, w]^\top$ and 3D space points $[x_w, y_w, z_w, 1]^\top$ in homogeneous coordinates:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \boldsymbol{K}\ [\boldsymbol{R}\ \boldsymbol{t}] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} [\boldsymbol{R}\ \boldsymbol{t}] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}. \tag{4.24}$$

here the subscript w represents the coordinates in the world coordinate frame.

In OpenGL this projection process is realized by two consecutive transform functions: glViewPort(x,y,$width$,$height$) and gluPerspective($fovy$, $aspect$,$zNear$,$zFar$). The two matrices generated from these two functions are as follows,

$$\boldsymbol{A} = \begin{bmatrix} \frac{width}{2} & 0 & \frac{width}{2} + x \\ 0 & \frac{height}{2} & \frac{height}{2} + y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.25}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\cot(\frac{fovy}{2})}{aspect} & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{4.26}$$

To be mentioned, the view direction of the real camera ($+Z$ direction) is opposite to the virtual camera's view direction ($-Z$ direction). To make them consistent, we

multiply an adjust matrix in the OpenGL CG pipeline as follows:

$$\begin{aligned}\begin{bmatrix} u \\ v \\ w \end{bmatrix} &= \boldsymbol{AB} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} [\boldsymbol{R}\ \boldsymbol{t}] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{width}{2} & 0 & \frac{width}{2}+x \\ 0 & \frac{height}{2} & \frac{height}{2}+y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\cot(\frac{fovy}{2})}{aspect} & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} [\boldsymbol{R}\ \boldsymbol{t}] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}.\end{aligned} \tag{4.27}$$

An equivalent relationship can be obtained from the projections in (4.24) and (4.27),

$$\begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{width\ \cot(\frac{fovy}{2})}{2\ aspect} & 0 & \frac{width}{2}+x \\ 0 & \frac{height\ \cot(\frac{fovy}{2})}{2} & \frac{height}{2}+y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.28}$$

Consequently, the parameters needed in the two OpenGL functions can be solved from (4.28) as

$$\begin{aligned} x &= u_0 - \tfrac{width}{2}, \\ y &= v_0 - \tfrac{height}{2}, \\ fovy &= 2\arctan 2(height, 2f_v), \\ aspect &= \tfrac{width\ f_v}{height\ f_u}. \end{aligned} \tag{4.29}$$

The viewport in OpenGL is set to $(width, height)$ pixels. The $zNear$ and $zFar$ parameters of the frustum are set to $zNear = 0.1$ and $zFar = 100$. Until now the simulation of a virtual camera with the real camera parameters has been completed.

To simulate the real pose $\boldsymbol{P}$ between the object and the camera, decomposition of

$\boldsymbol{P}$ is carried out to get the translation t_x, t_y, t_z and three Euler angles α, β, γ, which are respectively the rotation angles about the x, y and z axis. We use (tx, ty, tz) and (alpha, beta, gamma) to express the variables (t_x, t_y, t_z) and (α, β, γ) in the OpenGL code. Therefore, the OpenGL code should be realized as follows:

```
glViewPort(x,y,width,height);
glMatrixMode(GL_PROJECTION);
gluPerspective(fovy,aspect,zNear,zFar);
gluLookAt(0,0,0,0,0,1,0,1,0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(tx,ty,tz);
glRotatef(gamma,0,0,1);
glRotatef(beta,0,1,0);
glRotatef(alpha,1,0,0);
```

4.3.3 Implementation of Fast Search Application

The previous search for maximum similarity strategy runs the "render-compare" loop iteratively. We have adopted several acceleration methods to reduce the total search time so that a fast search application can be realized. The total time T_{total} equals to the product of iteration number N and the running time Δt in one "render-compare" loop,

$$T_{\text{total}} = N \times \Delta t. \tag{4.30}$$

Consequently, we have proposed two strategies to reduce the total search time.

(1) Reduce the number of search iterations (N). This step is realized by

Table 4.1: Comparison of Processing Time

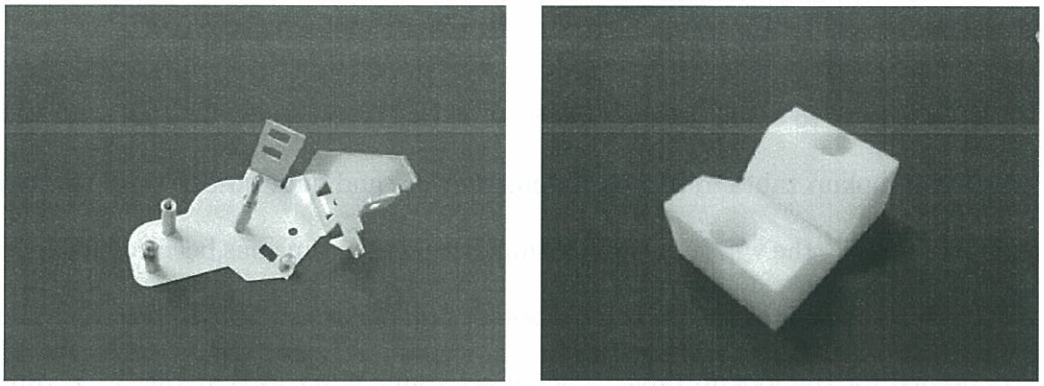
Image Size	Processing Time (ms)
1280×960	31.7
960×720	14.6
640×480	7.45
320×240	3.26
160×120	2.86
80×60	2.42

using a lookup table to replace the "render-compare" computation. The detail has been shown in section 4.2.2.

(2) Reduce the time in each iteration (Δt). This is realized by using a smaller image size after the image resizing operation. A comparison of the processing time for a loop with different image size is shown in Table 4.1. In the offline learning step VGA size (640×480) CG images are used to calculate the image features. When the "render-compare" loop runs on each candidate pose, a smaller size of 320×240 is used to calculate the SSD value between the CG images and real image.

4.4 Evaluation Experiments

Experiments have been carried out to evaluate our proposed coarse-to-fine strategy. The first part describes various experiments to evaluate the coarse step for pose estimation. The second part shows the applications of our estimation strategy by local optimization techniques. The third part lists the accuracy of our pose estimation strategy. The fourth part describes the final step of visual servo to place the camera relative to 3D objects. In the final part, we present two practical applications with the pose estimation: visual inspection of 3D objects and picking up a 3D object.



(a) Gear base (b) V-shaped work guide

Figure 4.8: Two objects used in the experiments: a gear base and a resin V-shaped work guide (*WGVN35* from *MISUMI* company)

4.4.1 Hardware and Software Configuration

The system is composed of a six DOF robot (*EPSON Pro Six PS5*), a desktop with Intel Core i7-920 2.67GHz, 3GB RAM, and IEEE 1394 cameras (*Grasshopper GRAS-03K2M/C*). The output image size from the camera is 640×480. Two objects are used in the experiments (Figure 4.8), a gear base and a resin V-shaped work guide (*WGVN35* from the *MISUMI* company). The CAD model file of the work guide is downloaded from the *MISUMI* website and converted into a VRML file. The CAD model of the gear base is created in *SolidWorks 2010* and converted into a VRML file. The VRML files are of version 2.0 and loaded into OpenGL to render a CG image of the objects. In the experiments we always use the same camera parameters. By calibrating the camera intrinsic parameters and extrinsic parameters, 1 mm in the X and Y direction of the camera frame approximately takes about 2.05 pixel in the u and v directions of the image.

4.4.2 Part I: Coarse Step Efficiency Evaluation

This part is to evaluate the efficiency of the coarse step. In this step we have adopted a lookup table to reduce the number of candidate poses. With the reduced number of poses, the whole maximum similarity search time has been reduced. Experiments have been done on the comparison of number of candidate poses and memory size needed for the lookup table. The object frame is set as that the X, Y direction is in the workbench plane and the Z axis is upright. The total number of poses in the search range is 497664.

(a) Comparison of the Number of Candidate Poses

Experiments for 6 DOF estimation of $(x, y, z, \alpha, \beta, \gamma)$ have been carried out to evaluate our PCA based lookup table method. The offline learning is done respectively for each object. The definition of image features $(\bar{u}, \bar{v}, \theta, A)$ is the same as in section 4.2.2. Three cases of experiments have been done with two objects.

- Use three features $(\bar{u}, \bar{v}, \theta)$ to calculate a table index without PCA processing;
- Use four features $(\bar{u}, \bar{v}, \theta, A)$ to calculate a table index without PCA processing;
- Use three PCA projected features (x_1, x_2, x_3) to calculate a table index. The features (x_1, x_2, x_3) are projected from the features $(\bar{u}, \bar{v}, \theta, A)$.

In the experiments, 10 poses are randomly chosen within the training ranges of each parameter. Then 10 CG images are rendered from the poses and used as the target images for pose estimation. As we use the "render-compare" strategy in the group of candidate poses, the running time for a search loop is proportional to the number of candidate poses. The search radius $(\Delta i, \Delta j, \Delta k, \Delta m)$ is determined in the experiments. Here we take the 4-feature case to describe this process in detail. First

Table 4.2: Number of the Candidate Poses with the Gear Base

Index	with $(\bar{u}, \bar{v}, \theta)$	with $(\bar{u}, \bar{v}, \theta, A)$	with (x_1, x_2, x_3)
1	1773	702	563
2	3658	1586	1525
3	2833	1276	1262
4	1165	579	689
5	1564	566	814
6	3998	1098	1541
7	1299	400	384
8	1331	314	480
9	977	51	503
10	1271	267	590
Mean	1986.9	683.9	835.1

Table 4.3: Number of the Candidate Poses with the Work Guide

Index	with $(\bar{u}, \bar{v}, \theta)$	with $(\bar{u}, \bar{v}, \theta, A)$	with (x_1, x_2, x_3)
1	2086	479	878
2	1612	370	842
3	1142	793	699
4	719	623	915
5	1770	570	811
6	2997	732	983
7	2188	400	623
8	241	196	180
9	179	59	53
10	432	248	424
Mean	1336.6	447.0	640.8

we choose a large search radius to test with the 10 random input poses. For each test pose P_n^*, $(n = 1, 2...10)$ the index difference $(\delta i_n, \delta j_n, \delta k_n, \delta m_n)$ between the list containing the optimum pose and the list index (i^*, j^*, k^*, m^*) is calculated. Then after the 10 times test, the search radius is set as the same as the maximum radius in each dimension of the 10 test results $(\delta i_n, \delta j_n, \delta k_n, \delta m_n)$, $n = 1, 2...10$.

The number of candidate poses in the 10 times test (test index is from 1 to 10) is listed in Table 4.2 (for gear base) and Table 4.3 (for V-type work guide). For

the "render-compare" loop, we adopt a smaller size of 320 × 240 which takes about 3.26 ms in the experiment. By comparing the results of "with 3-feature" and "with 4-feature" in the two tables, we can find that the number of poses is reduced by using more features, which means using more features can effectively reduce the number of candidate poses. This reduction can be obtained from the fact that:

- Searching with 3 features $(\bar{u}, \bar{v}, \theta)$ means finding a group of poses from which the CG images' features fall in the range of $(u^* \pm \Delta u, v^* \pm \Delta v, \theta^* \pm \Delta\theta)$. Here (u^*, v^*, θ^*) are the features from the real image.

- Searching with 4 features $(\bar{u}, \bar{v}, \theta, A)$ means finding a group of poses from which the CG images' features fall in the range of $(u^* \pm \Delta u, v^* \pm \Delta v, \theta^* \pm \Delta\theta, A^* \pm \Delta A)$. As $(\Delta u, \Delta v, \Delta\theta)$ keep same in two cases, it is obviously that the candidate poses in this case belong to a subgroup of the group of poses in the 3-feature case.

Meanwhile from the comparison of PCA method with 4-feature methods, both have similar performances as the numbers in both methods are similar.

(b) Comparison of the Memory Size

Previous part shows using lookup table with more features can reduce the number of candidate poses. However, using more features causes another problem of the required memory. The memory is determined by two items in the lookup table implementation.

(**1**): For each pose a structure of 8 bytes is required to fill in the lookup table, where the structure is defined as

```
struct pose {
long index; //To store the index of the pose
```

Table 4.4: The Size of the Lookup Tables

Object	with $(\bar{u}, \bar{v}, \theta)$	with $(\bar{u}, \bar{v}, \theta, A)$	with (x_1, x_2, x_3)
Gear Base	$127 \times 116 \times 360$ =5,303,520	$127 \times 116 \times 360 \times 50$ =265,176,000	$364 \times 131 \times 154$ =7,343,336
Work guide	$71 \times 68 \times 360$ =1,738,080	$71 \times 68 \times 360 \times 50$ =86,904,000	$361 \times 94 \times 95$ =3,223,730

```
struct pose *next;//To connect next element in the list
```

}

One such structure in our system takes 8 bytes memory, therefore the memory required in this part is $N \times 8$ bytes, where N is the number of total poses. In the experiments we use $N = 497664$, hence the total memory needed is 3.88 MB.

(**2**): For each element of a list in the lookup table, two pointers are required for marking the list head and current element in the list. In our system (Windows XP 32 bit) a pointer takes 4 bytes. Hence the total memory needed in this part is $TableSize \times 8$ bytes, where $TableSize$ is the product of each dimension in the lookup table. The size of the lookup table in the previous three cases are shown in Table 4.4.

From the table we can find that though using more features (four) can reduce the number of candidate poses, the required memory also increases rapidly. For instance, in the gear base experiments, the memory needed in the 3-feature and 4-feature methods is $5303520 \times 8 = 40.46$ MB and $265176000 \times 8 = 2023$ MB respectively. Meanwhile, by using the PCA projection to reduce the feature dimension to 3, though the three new features (x_1, x_2, x_3) are calculated from the four features $(\bar{u}, \bar{v}, \theta, A)$, the memory required for the 3 dimensional PCA method is $7343336 \times 8 = 56.03$ MB, much smaller than in the 4-feature methods.

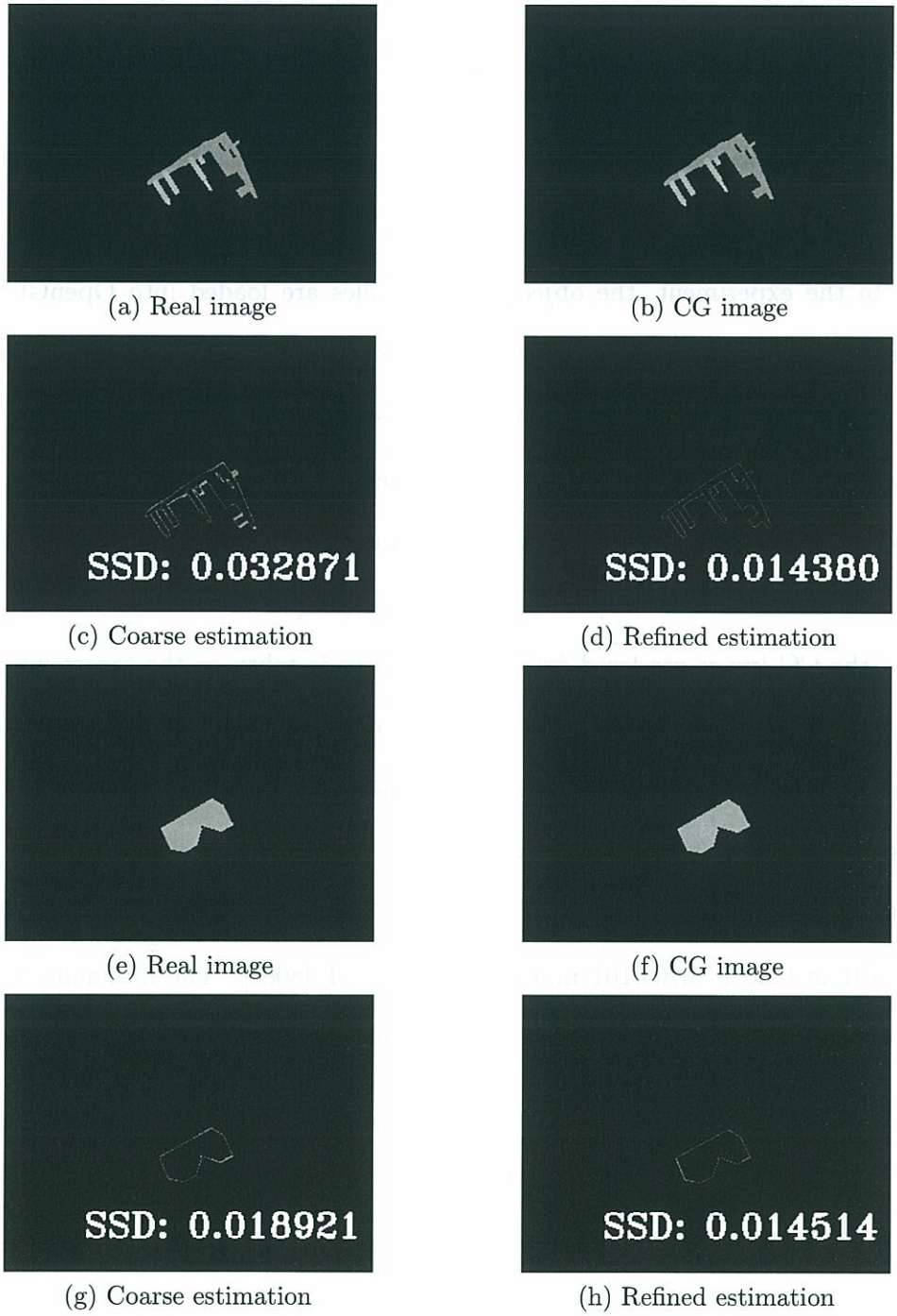
In the experiments, the search radius $(\Delta i, \Delta j, \Delta k)$ in the search cell of the PCA method (with the gear base) is $(10, 7, 3)$. The number of cell is $7343336/(20 \times 14 \times 6) \sim$

4371. While in the 4-feature method, the search radius $(\Delta i, \Delta j, \Delta k, \Delta m)$ of the cell is $6, 6, 10, 3$, the number of cell is $265176000/(12 \times 12 \times 20 \times 6 \sim 15346$. We have shown that the average number of poses in the 4-feature and PCA methods are similar (683.9 and 835.1). As the total number of poses are the same (N=497664), this shows that the distribution of features have been improved in the PCA methods (the distribution in the 4-feature lookup table is more sparse than in the 3-feature lookup table). Therefore, this experiment shows that our PCA projection method can effectively reduce the memory required for the lookup table while keeping the similar number of poses as in more features method. Compared with directly using features in the lookup table, using PCA projection can ensure a larger number of total poses with the same memory size. This increase means we can make the lookup table with a larger training range in each parameter of the poses with the same training resolution, which also means our system can find an initial pose in a larger range.

(c) Images of the Experiments

This section shows the image sequences of the coarse step in the experiments. As mentioned in section 4.3.1, after finding a pose within the lookup table, a small group of poses around the coarse estimation are calculated and the "render-compare" loop is adopted on these poses to estimate a more accurate solution.

The images of pose estimation with 6 DOF parameters have been shown in Figure 4.9. From the decrease of the SSD values between the input image and CG image rendered from the estimated pose, it has shown that this neighbor search step has increased the estimation accuracy of the coarse step.



(a) Real image (b) CG image

(c) Coarse estimation (d) Refined estimation

(e) Real image (f) CG image

(g) Coarse estimation (h) Refined estimation

Figure 4.9: Pose estimation of the the gear base and V-type work guide. Input image is rendered from a random pose. From the decrease of SSD values, image (d) and (h) have shown that the pose estimation has been refined.

4.4.3 Part II: Fine Step Efficiency Evaluation

This part is to evaluate the efficiency of the fine step in our coarse-to-fine strategy. We have realized the local minimization techniques in section 4.2.4 to refine the pose estimation.

In the experiment, the objects' VRML files are loaded into OpenGl to render the CG images. A simple 6D vector is adopted to describe the camera pose as $\boldsymbol{P} = [x, y, z, \alpha, \beta, \gamma]$, where x, y and z represent the translations in the X, Y and Z directions while α, β and γ describe the three rotation angles about the X, Y and Z axes. To show the pose estimation accuracy, the virtual object is moved to a new pose with transformation $[dx, dy, dz, d\alpha, d\beta, d\gamma] = [0.01 \text{ m}, 0.01 \text{ m}, 0.01 \text{ m}, 2°, 2°, 2°]$ and the CG image rendered from the new pose is taken as the target image. Then the pose estimation approach tries to estimate the transformation so that it can render a same CG image as the target image. With local minimization method with Gauss Newton optimization, we have got the estimation $[dx^*, dy^*, dz^*, d\alpha^*, d\beta^*, d\gamma^*]$ of $[0.0101 \text{ m}, 0.0098 \text{ m}, 0.0100 \text{ m}, 2.0057°, 1.9947°, 1.9636°]$ for the gear base object and $[0.0101 \text{ m}, 0.0098 \text{ m}, 0.0101 \text{ m}, 1.7385°, 1.5920°, 1.3808°]$. The minimum SSD values are 0.0040 and 0.0057 in the two cases. The image sequence of the two objects is shown in Figure 4.11.

To evaluate the coarse-to-fine strategy, we have compared the accuracy of solutions from the coarse step and from the fine step. The experiment is carried on the gear base with 6 DOF estimation. The test images are rendered from 10 random poses. The test poses are compared with the estimation from the coarse step and fine step respectively. The comparison results are drawn in Figure 4.13. From the results we can find that the fine step has improved the accuracy of the solution from the coarse

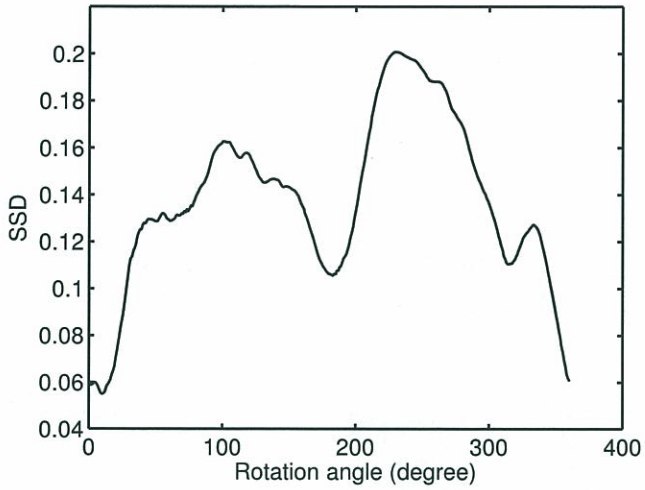
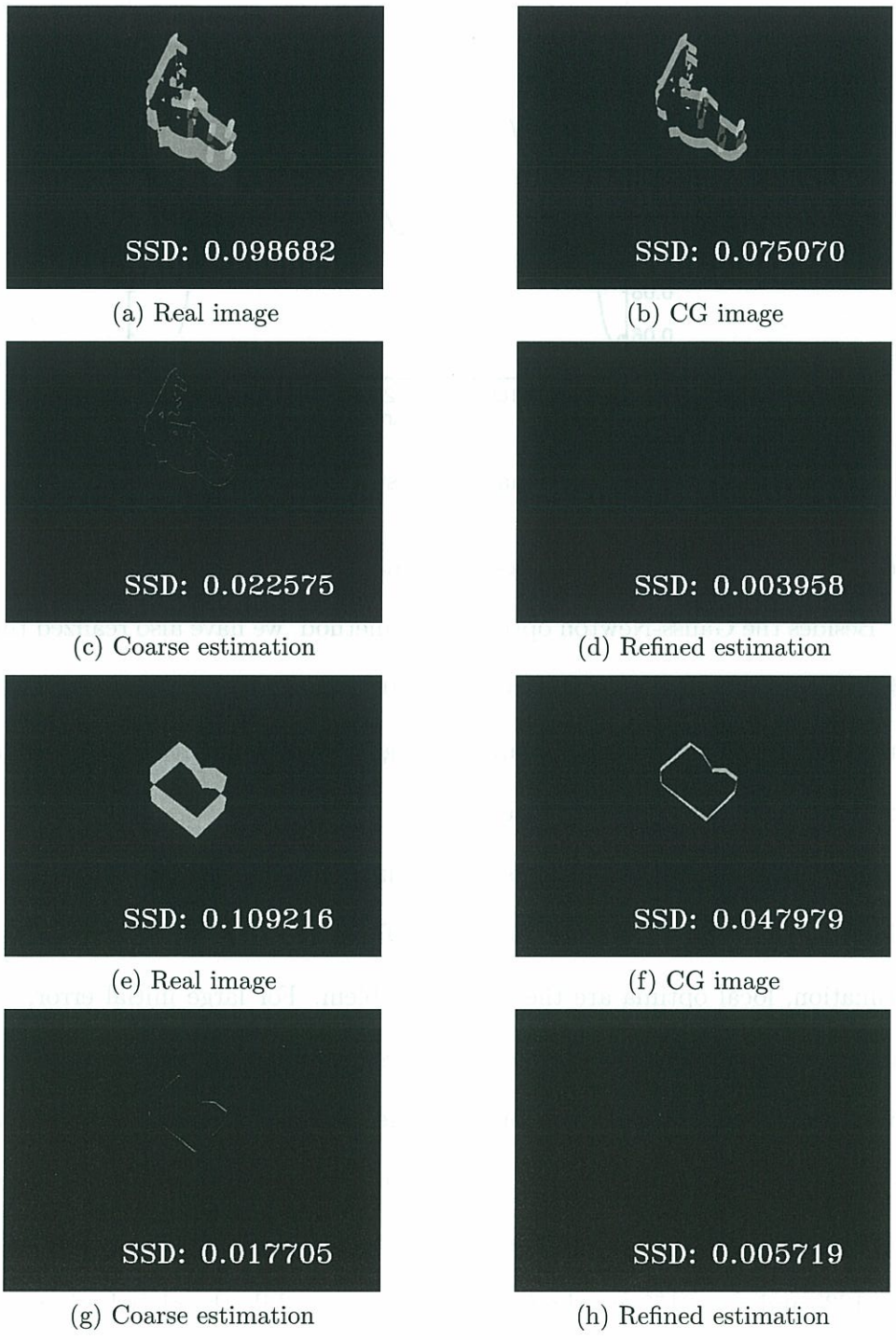


Figure 4.10: SSD values (normalized) respecting to the rotation γ in [0°, 360°]

step. By this means, our coarse-to-fine strategy has been evaluated.

Besides the Gauss-Newton optimization method, we have also realized the moving edge algorithm by Drummond and Cipolla [94] for pose estimation of the V-type work guide, which has a regular shape and the edges can be extracted from the image. One image sequence is shown in Figure 4.12.

To be mentioned, due to the local optimization nature of these algorithms, a good initialization point is important for the performances of the algorithms. In the pose estimation, local optima are the critical problem. For large initial error, the Gauss Newton method fails to estimate the right solution instead it converged to its local minimum. The figure 4.10 shows one example. The CG images are rendered according to a rotation about the Z axis for one round and the SSD values between the CG images and a target image have been calculated and shown in the figure. In the figure, $\gamma = 183°$ and $\gamma = 315°$ are the local optima, meanwhile the global minimum SSD is at $\gamma = 10°$. Therefore, due to the this limit, we have to use the coarse step with a global search to avoid this local optima problem.



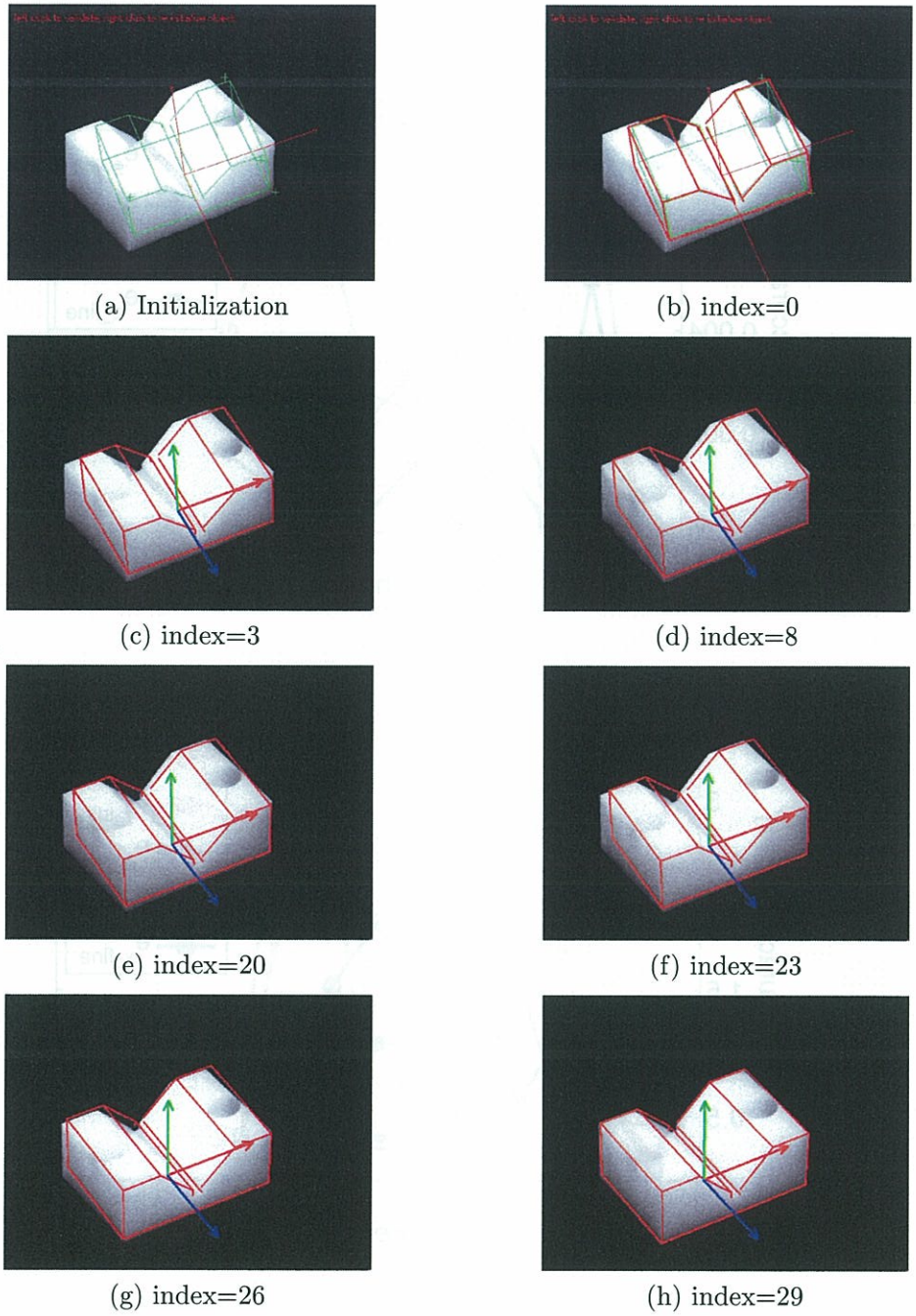
(a) Real image (b) CG image

(c) Coarse estimation (d) Refined estimation

(e) Real image (f) CG image

(g) Coarse estimation (h) Refined estimation

Figure 4.11: Pose estimation with local optimization method.The decrement of the image difference (normalized SSD value) shows the pose estimation converges and reliable estimation can be obtained.



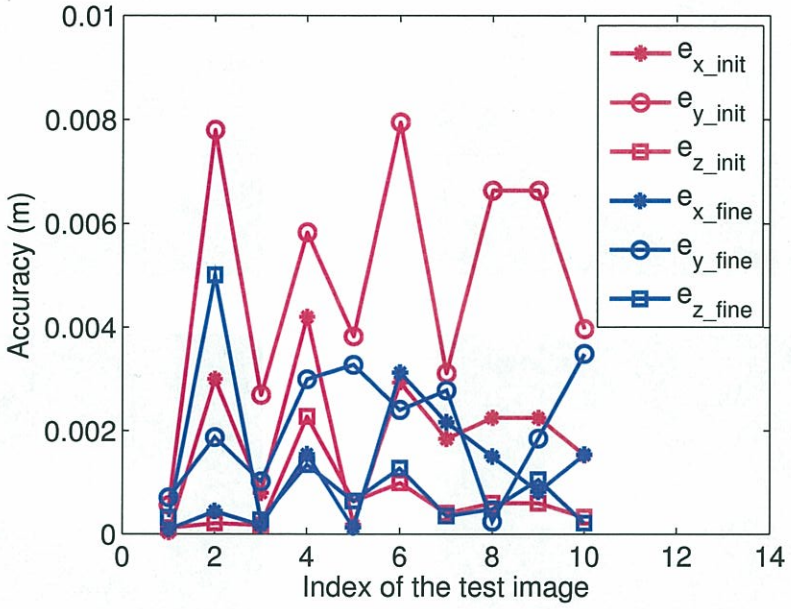
(a) Initialization (b) index=0

(c) index=3 (d) index=8

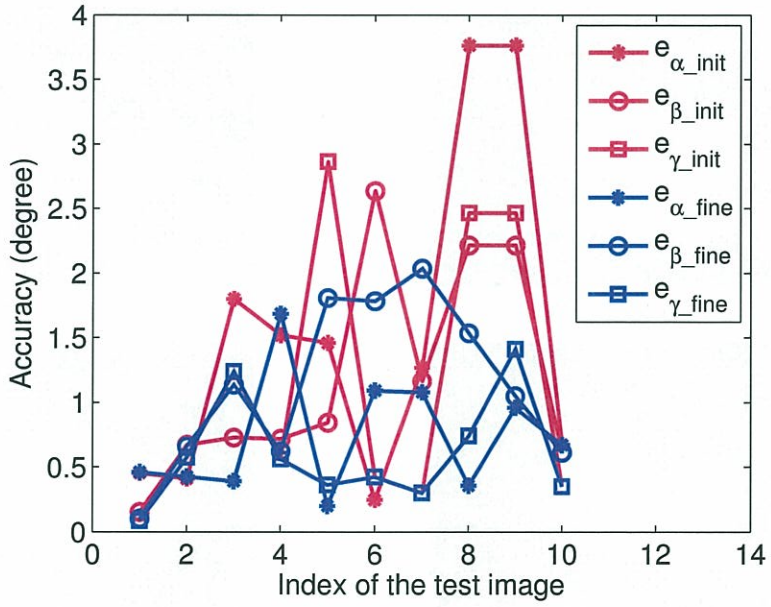
(e) index=20 (f) index=23

(g) index=26 (h) index=29

Figure 4.12: Pose estimation with moving edge algorithm.The initialization is manually realized and the object's wire frame model is shown as the green lines. The projected edges with the estimated pose are shown in red lines. The convergence of the red lines to the real object's edges shows reliable pose estimation can be obtained.



(a)



(b)

Figure 4.13: Accuracy evaluation of the 6 DOF estimation case. The errors between the estimated parameters and input ones are shown in the figure with (a) position error, (b): orientation error. The red data stands for the results from the initial coarse step, while the blue data denotes the final results from the fine step with local optimization techniques.

Table 4.5: Accuracy of the Pose Estimation in 6 DOF Cases

Parameter	Gear base Maximum Error	Gear base RMS Error	V-shaped guide Maximum Error	V-shaped guide RMS Error
x	0.0042 m	0.0022 m	0.0042 m	0.0018 m
y	0.0079 m	0.0061 m	0.0108 m	0.0046 m
z	0.0023 m	0.0008 m	0.0023 m	0.0009 m
α	3.7620°	1.9609°	6.5488°	3.7919°
β	2.6336°	1.4377°	5.1699°	2.6643°
γ	2.8604°	1.5124°	7.4156°	4.0355°

4.4.4 Part III: Pose Estimation Accuracy

With the coarse-to-fine strategy, we have evaluated the accuracy of the whole system with the gear base and V-shaped work guide. The lookup table is trained with resolution of 5 mm in translation and 5° in rotation. The test is carried out with 10 random poses. The result is shown in Table 4.5.

4.4.5 Part IV: Visual Servo with a Complex-shaped Object

With the pose solution from the previous steps, the transformation of the object can be calculated. Due to the estimation error of the ${}^{b}M_{o}$ in equation (4.5) (manually measured), the new trajectory can not ensure an optimum one. Therefore, visual servo is adopted in our framework to adjust the camera. In our applications when the camera is close to the objects, the small part of the objects can be treated as planar. Therefore, we can use homography-based visual servo to accurately place the camera. As the difference between the new trajectory and the final desire pose is small, this method can work effectively. We adopt the homography-based visual servo method in section 2.2. One image sequence is shown in Figure 4.14.

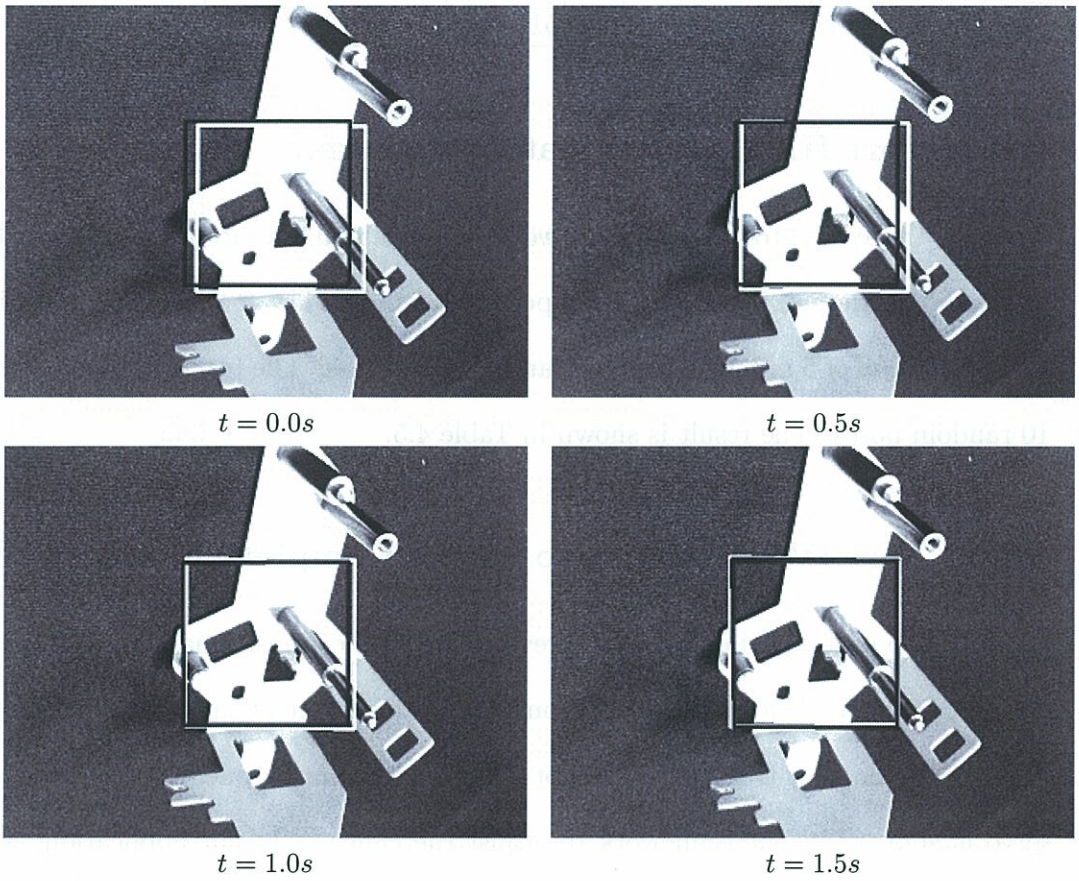


Figure 4.14: Visual Servo. The white and black quadrilateral boxes describe the reference and current positions of the tracking area separately. The image at $t = 1.5s$ shows that the visual servo scheme can work well (both boxes coincide finally).

4.4.6 Part V: Applications with the Object Pose Estimation

This part introduces two practical applications with our pose estimation strategy. The objects in the experiments are the gear base shown in previous parts. The object's pose is estimated by our combination strategy in the beginning. In the first application, this pose information is used to move an inspection camera to check this gear base, the second application uses the pose information to move a robot hand to pick up this base. The detail is introduced in the rest parts.

In the visual inspection applications, the system setup is shown in Figure 4.15. The system is composed of a six DOF robot (*EPSON Pro Six PS5*), and two IEEE 1394 cameras (*Grasshopper GRAS-03K2M/C*). The inspection camera is mounted on the robot while the environment camera is used for the pose estimation. The environment camera is necessary because the inspection camera's field of view is chosen for inspecting a small region in detail, which is not suitable for the whole object. In this experiment, we use one good part and one faulty part (a small part is bent intentionally) to test the system (Figure 4.16).

After placing the camera to its desired poses, the inspection task is realized by a template matching approach (Figure 4.17) which is based on the Normalized Correlation Coefficient (NCC)

$$R(u,v) = \frac{\sum_{u',v'} \left(T(u',v')I(u+u',v+v')\right)}{\sqrt{\sum_{u',v'} T(u',v')^2 \sum_{u',v'} I(u+u',v+v')^2}}, \tag{4.31}$$

where u', v' are the pixel coordinates in the template patch T. u and v are the top left pixel coordinates of the current convolution region in the current image. By realizing this NCC convolution operation on the real image with the template patch,

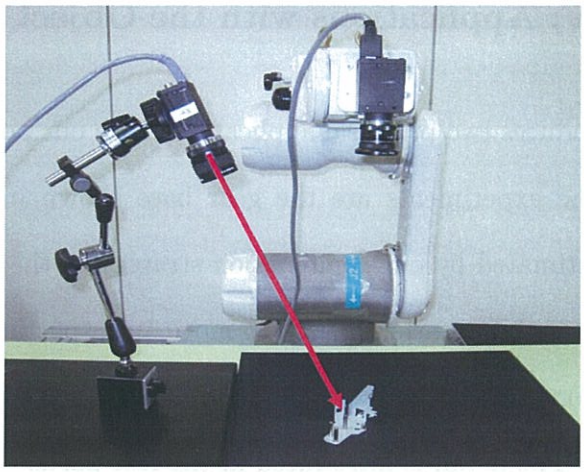
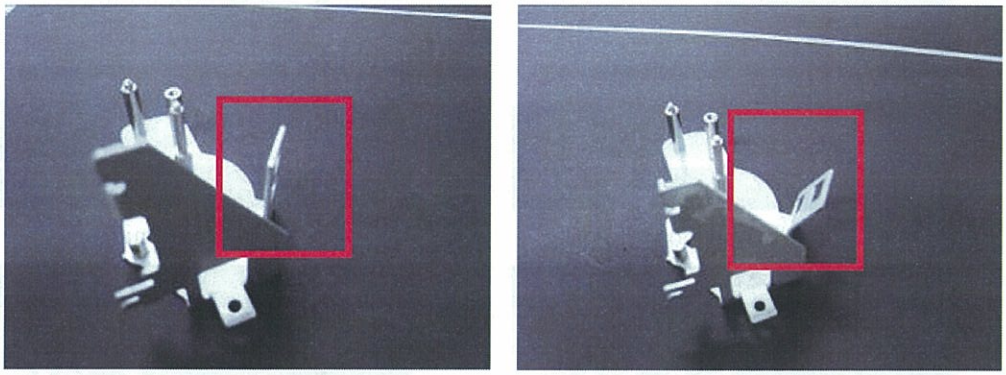


Figure 4.15: The visual inspection system is composed of a six DOF robot (*EPSON Pro Six PS5*) and two IEEE 1394 cameras (*Grasshopper GRAS-03K2M/C*). The inspection camera is mounted on the robot while the environment camera is used for the pose estimation.

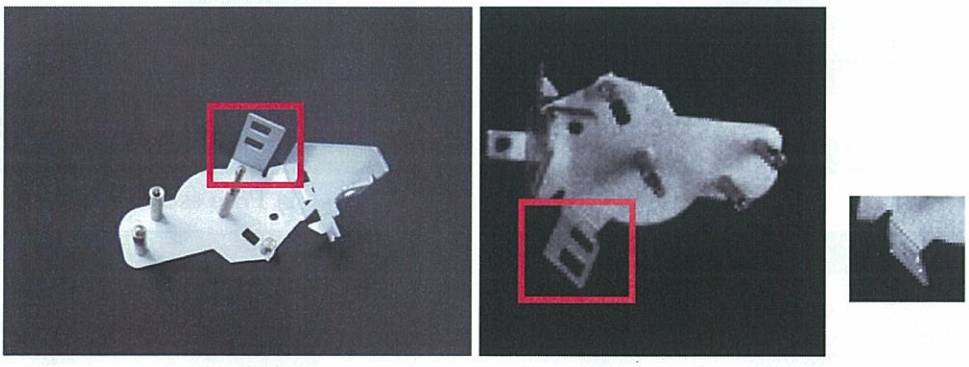
a maximum NCC value $R(u, v)$ can be found. If it is above a preset threshold (0.95 in our application), the inspection point is treated as good. Otherwise, the object will be treated as a faulty part.

An image sequence extracted from the experiment video is shown in 4.18. First the good object is put at a random pose (a), then its pose is estimated (b). With this pose estimation, the desired pose of the camera is calculated and the robot is moved to this new pose (c). The template matching runs in (d) and the result is good (shown with green marker). For the NG part, its initial pose (e) is estimated in (f). Though there is some difference between the real object and the CAD model, our combination strategy still can obtain the pose estimation. Then the camera is moved to the new pose with the pose information (g) and the template matching runs in (h) and gives the result of "NG" with red marker in the image. From this experiment, the efficiency of our camera positioning in the visual inspection system has been evaluated.



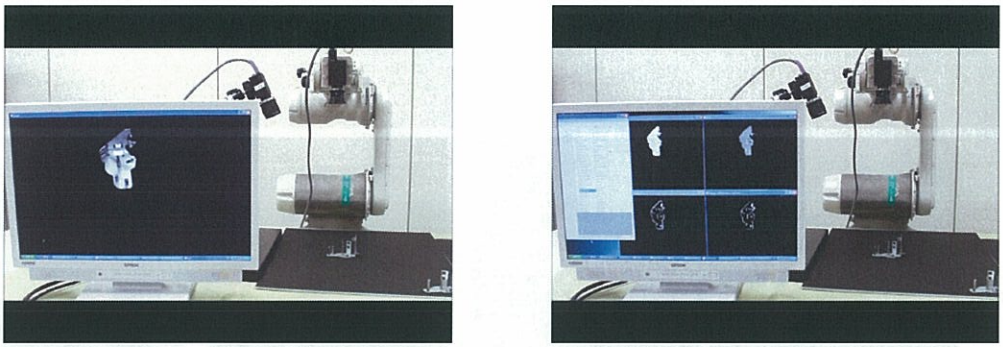
(a) Good (b) NG

Figure 4.16: One good part and one faulty (NG) part with a small part bent intentionally are used in the visual inspection application.



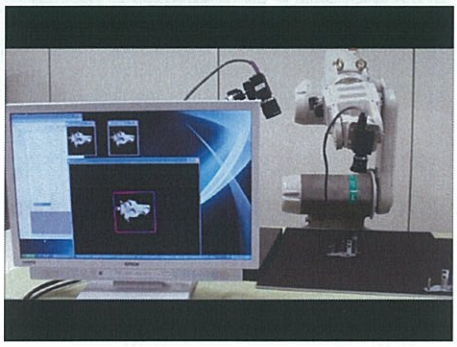
(a) Object (b) Image (c) Template

Figure 4.17: The template matching is used to determine whether the part is good or faulty. The red rectangular marks the region which needs to be inspected.



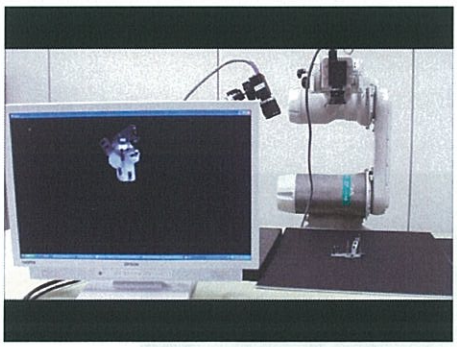
(a) Initial pose

(b) Pose estimation

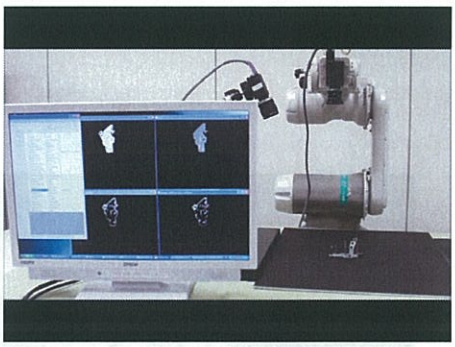


(c) Move with the pose

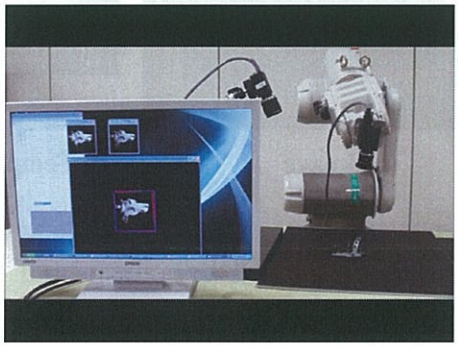
(d) Template matching



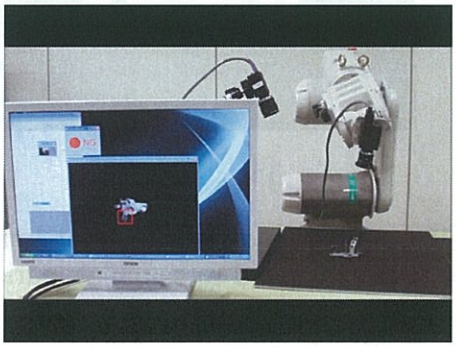
(e) Initial pose



(f) Pose estimation



(g) Move with the pose



(h) Template matching

Figure 4.18: The whole process for the good part is shown from (a) to (d), while the process for the NG part is shown from (e) to (h).

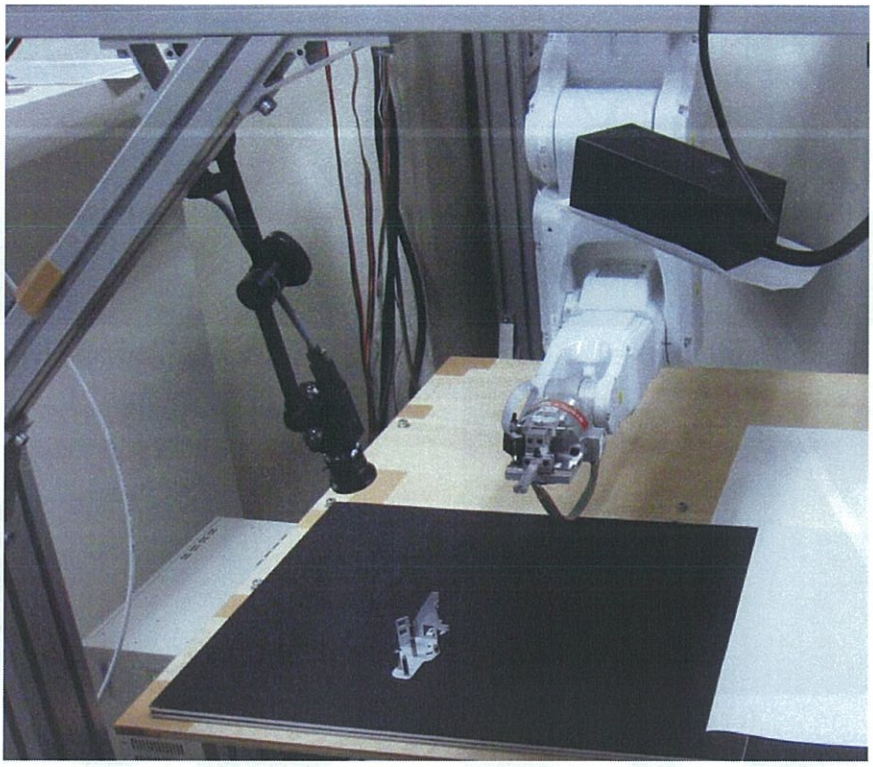
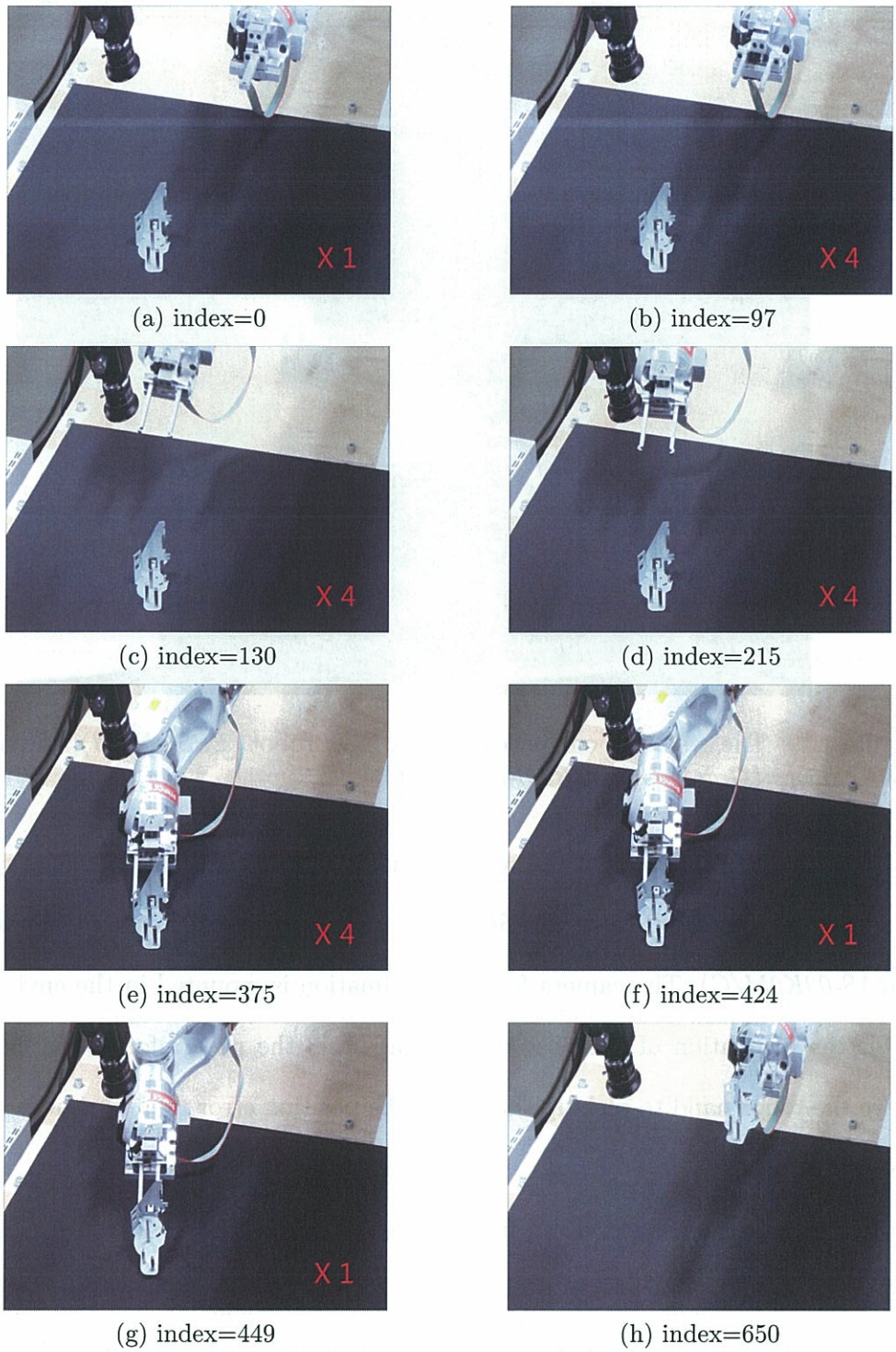


Figure 4.19: The system is composed of a six DOF robot (*EPSON C3*), and an IEEE 1394 camera (*Grasshopper GRAS-03K2M/C*)

In the second application, the system setup is shown in Figure 4.19. The system is composed of a six DOF robot (*EPSON C3*), and an IEEE 1394 camera (*Grasshopper GRAS-03K2M/C*). The camera for pose estimation is mounted in the environment. After the estimation of the pose of the gear base, the pose information is used to move the robot hand to pick up the base. The position error requirement is 1 mm in the experiment.

An image sequence extracted from the experiment video is shown in 4.20. First the good object is put at a random pose (a), then its pose is estimated. With this pose estimation, the robot hand is moved to pick up the object ((b) to (e)). Image (f) to (h) shows that the robot hand has successfully picked up the object. From this experiment, the efficiency of our pose estimation has been evaluated.



(a) index=0 (b) index=97

(c) index=130 (d) index=215

(e) index=375 (f) index=424

(g) index=449 (h) index=650

Figure 4.20: An image sequence extracted from the video of picking up a gear base with its pose estimation. (a): initial case, (b): the robot hand starts moving with the estimated pose, (c) to (e); the robot hand continues moving to the object, (f) to (h): the robot hand holds the object and picks it up.

4.5 Conclusions

In this chapter we have proposed a coarse-to-fine framework to realize camera positioning relative to 3D objects. In the framework, the coarse step provides initial pose estimation within large initial pose ranges by a global search for the maximum similarity between the real image and those CG images rendered in OpenGL. The fine step refines the pose estimation by local optimization approaches and move the camera to its desired pose. The final visual servo helps to move the camera to its desired pose. Various experiments have been carried out on different kinds of objects. The results show that our framework can efficiently provide a robust pose solution and use it to place a camera with complex 3D objects for visual inspection applications.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this dissertation, we have proposed two combination strategies respectively to place a camera with planar objects or 3D objects in visual inspection applications.

In chapter 2 we have proposed a GPU accelerated ESM algorithm to address the need for faster homography solutions in the homography-based visual servo. With carefully parallelization and optimization for the ESM algorithm in CUDA, our GPU version ESM algorithm can work at about $20-30$ times faster than its CPU version. Accordingly, the system can run at high speed to track fast moving objects which were very difficult to track in previous CPU-only systems. Besides the ESM algorithm, we extend the optimization work to the Thin-Plate Spline (TPS) deformable image registration. We present the evaluation experiments to verify the efficiency of our optimization techniques. The evaluation experiments show that, compared with the CPU-ESM implementation, our GPU implementation has accelerated it by 20-30 times and can realize the object tracking at high speed. The extension of the GPU

optimization to the deformable image registration problem also shows that our GPU acceleration implementation is very useful for fast visual tracking applications.

In chapter 3 we focus on solving the camera positioning task by a homography-based visual servo scheme with planar objects. We have proposed a combination strategy of ESM tracking algorithm and SIFT matching algorithm for homography estimation. Both algorithm can find homography estimation independently, however, the ESM tracking algorithm needs good initialization, while the SIFT matching algorithm can provide a homography estimation from a large region but it is slow. Therefore, in the combination we have adopted the multi-thread programming technique to run both GPU-ESM and GPU-SIFT algorithms on separate GPUs simultaneously and proposed a switch strategy between both methods based on the ZNCC value. Evaluation experiments show that, compared with the ESM or SIFT method, our combination strategy has improved the system performances on processing speed, convergence range and the robustness to occlusions. Therefore, the system can find homography estimation from large initial pose ranges at a high speed and robust to occlusions. The real visual inspection application shows that our combination strategy is very useful for the homography-based visual tracking and visual servo applications.

In chapter 4 we have proposed a coarse-to-fine strategy to place a camera with respect to 3D objects. We have proposed an efficient view-based pose estimation method in the coarse step. The initial pose is estimated by a global search for the maximum similarity between the real image and those images rendered from computer graphics (CG) image, meanwhile the local optima as in other methods have be effectively avoided. Firstly, a series of image features are calculated from the image

moments of all the CG images rendered from all possible poses in the search space. Then the principal component analysis (PCA) is carried out on these image features to reduce the feature dimension to save memory space. With the reduced PCA projected features, a lookup table is created. By searching the maximum similarity in a much smaller space in the lookup table, a remarkable speedup of the pose estimation has been obtained. The evaluation experiments show that, compared with those feature based methods which require a good guess of the initial pose, our combination strategy can find the pose estimation in large range. Real applications of visual inspection and pick-and-place of 3D objects have shown that the combination strategy is an important contribution for positioning an inspection camera with 3D objects.

5.2 Future Work

Currently our applications of 3D object pose estimation are limited in those single objects case within a clear scene. For those cases with heavily cluttered background scenes, our current approaches does not work well. Therefore, our future work will concentrate on the 3D pose estimation in sophisticated environments, which are more challenging work to inspire us to do further study in this filed. And for the visual inspection applications, fast positioning and accurate positioning are both required. Therefore our future work also includes the camera trajectory optimization to speed up the camera traveling.

Appendix A

Perspective Projection

In this thesis we adopt the pinhole camera model for the perspective projection in the image formation process (Figure A.1). Given a point in the world coordinate frame $M = [x, y, z]^\top$ and its corresponding 2D image point $m = [u, v]^\top$, the projective mapping from world coordinates to pixel coordinates can be expressed by

$$s\widetilde{m} = \boldsymbol{P}\widetilde{M}, \tag{A.1}$$

where s is a scale factor, $\widetilde{m} = [u, v, 1]^\top$ and $\widetilde{M} = [x, y, z, 1]^\top$ are the homogeneous coordinates of points m and M, and $\boldsymbol{P}$ is a 3×4 projection matrix. The projection matrix $\boldsymbol{P}$ can be decomposed as multiplication by two matrices

$$\boldsymbol{P} = \boldsymbol{K}\ [\boldsymbol{R}\ \boldsymbol{t}]. \tag{A.2}$$

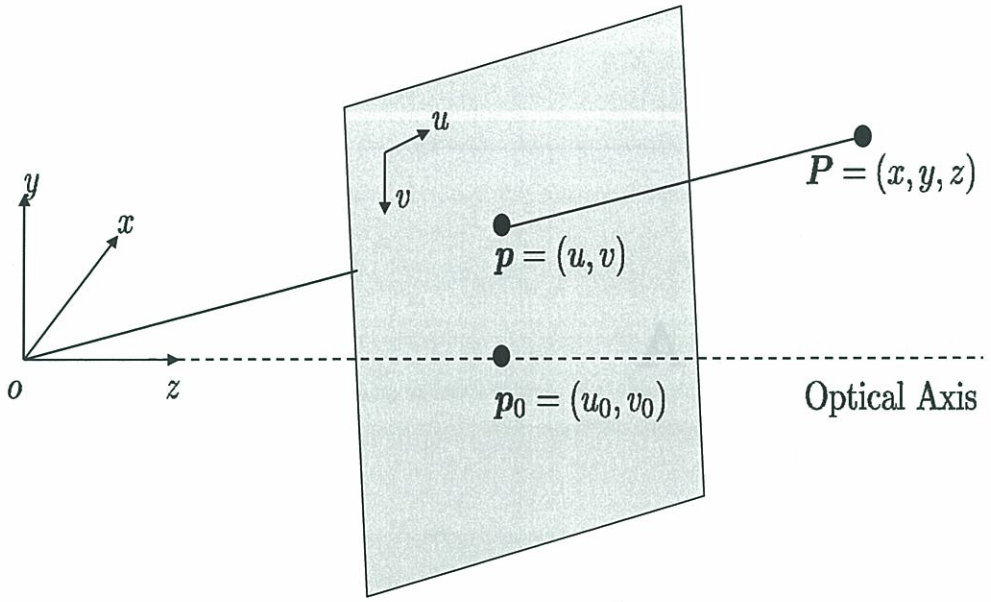


Figure A.1: A pinhole camera model is used to represent the perspective projection.

The matrix $\boldsymbol{K}$ is the 3×3 camera intrinsic parameters related to the camera internal parameters such as focal length. The $\boldsymbol{K}$ has the form as

$$\mathbf{K} = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3}$$

where f_u and f_v are the focal length in pixel dimensions, u_0 and v_0 are the pixel coordinates of the principal point $\boldsymbol{P}_0$, s is the screw. s is non-zero only if the u and v axes are not perpendicular, which is rare in modern cameras. Therefore, s is set to 0 in our applications.

Furthermore, the 3×4 matrix $[\boldsymbol{R}\ \boldsymbol{t}]$ refers to the camera extrinsic parameters which represent the Euclidean transformation from a world coordinate frame to a camera coordinate frame. It is composed of a 3×3 rotation matrix $\boldsymbol{R}$ and a 3×1

translation vector $\boldsymbol{t}$,

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} = \begin{bmatrix} R_{xx} & R_{yx} & R_{zx} & t_x \\ R_{xy} & R_{yy} & R_{zy} & t_y \\ R_{xz} & R_{yz} & R_{zz} & t_z \end{bmatrix}. \tag{A.4}$$

Appendix B

Pose Representation and Transformation

The matrix $[\boldsymbol{R}\ \boldsymbol{t}]$ in Eq. A.2 defines the orientation and the position of the camera relative to an object. We will refer to it as the *camera pose* in this thesis. Therefore, a pose can be interpreted in two distinct ways:

1. It represents a coordinate transformation between two different coordinate frames.

2. It gives the orientation and position of a coordinate frame with respect to another frame.

B.1 Basic Transformation

To describe the transformation with a matrix $[\boldsymbol{R}\ \boldsymbol{t}]$, a set of basic homogeneous transformation matrices are given. The basic transformation set is composed of three translation dx, dy, dz along the x, y, z axis and three rotation α, β, γ about the x, y, z

axis respectively:

$$
Trans(x,dx) = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Rot(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & -s\alpha & 0 \\ 0 & s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
Trans(y,dy) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Rot(y,\beta) = \begin{bmatrix} c\beta & 0 & s\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s\beta & 0 & c\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \text{(B.1)}
$$

$$
Trans(z,dz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Rot(z,\gamma) = \begin{bmatrix} c\gamma & -s\gamma & 0 & 0 \\ s\gamma & c\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

where $c\cdot$ and $s\cdot$ are the abbreviation form of $\cos\cdot$ and $\sin\cdot$ respectively. With this series of basic transformations, any transformation $[\boldsymbol{R}\ \boldsymbol{t}]$ in the 3D space can be decomposed as composition of several basic transformations.

B.2 Pose Transformations

In robot applications, when the object is moved relative to the camera, the relative pose between the camera and the object is updated by the object motion $[\boldsymbol{R}\ \boldsymbol{t}]$. Let $\boldsymbol{P_{new}} = [\boldsymbol{R}_{new}\ \boldsymbol{t}_{new}]$ and $\boldsymbol{P_{old}} = [\boldsymbol{R}_{old}\ \boldsymbol{t}_{old}]$ be the respective camera pose after and before the motion. According to different reference frame of the object motion $[\boldsymbol{R}\ \boldsymbol{t}]$ shown in Figure B.1, the relationship between these two poses are listed as follows :

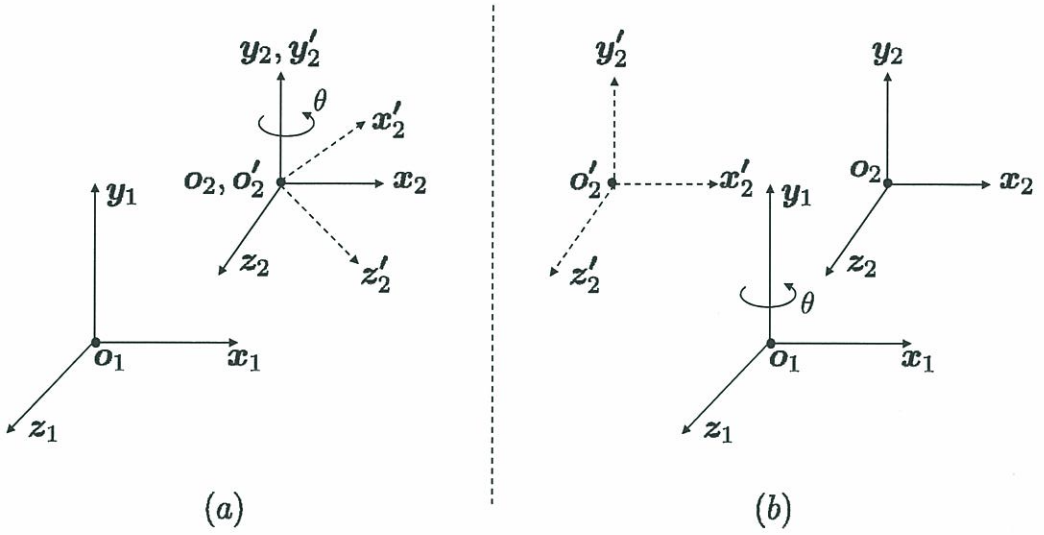


Figure B.1: Two instances to illustrate the different cases. (a): The rotation θ of frame $\sum_2$ is about its own $\boldsymbol{y}_2$ axis. (b): The rotation θ of frame $\sum_2$ is about the $\boldsymbol{y}_1$ axis of the frame $\sum_1$.

- Object motion $[\boldsymbol{R}\ \boldsymbol{t}]$ is expressed with respect to its own frame. For instance, case (a) in Figure B.1, the frame $\sum_2$ is rotated about its own axis. The new pose is calculated by **postmultiplying** the motion to the old pose,

$$\boldsymbol{P}_{new} = \boldsymbol{P}_{old}[\boldsymbol{R}\ \boldsymbol{t}]. \tag{B.2}$$

- Object motion $[\boldsymbol{R}\ \boldsymbol{t}]$ is expressed with respect to other fixed frame. For instance, the case (b) in Figure B.1, the frame $\sum_2$ is rotated about an axis of the frame $\sum_1$. In this case, the new pose is calculated by **pretmultiplying** the motion to the old pose,

$$\boldsymbol{P}_{new} = [\boldsymbol{R}\ \boldsymbol{t}]\boldsymbol{P}_{old}. \tag{B.3}$$

Following Eq. B.2 and Eq. B.3, the new pose ${}^1\boldsymbol{P}_2$ in the two cases of Figure B.1 can be calculated as

$$\begin{aligned} (a): {}^1\boldsymbol{P}_2' &= {}^1\boldsymbol{P}_2\ Rot(y,\theta), \\ (b): {}^1\boldsymbol{P}_2' &= Rot(y,\theta)\ {}^1\boldsymbol{P}_2. \end{aligned} \tag{B.4}$$

B.3 Pose Parameterization

The group of all poses (or rigid body motions) is known as the *Special Euclidean Group* and is denoted by $SE(3)$. An element of $SE(3)$ is of the form $(\boldsymbol{R}, \boldsymbol{t})$ where $\boldsymbol{R} \in SO(3)$ (Special Orthogonal Group) and $\boldsymbol{t} \in \mathbb{R}^3$. For estimation and numerical optimization purposes, the camera pose must be appropriately represented. While representing the translations is easy to realize by a simple 3×1 vector in $\mathbb{R}^3$, parameterizing rotation is more difficult to do it well. Several parameterization methods have been proposed to represent the rotation: Euler-angle representation, Roll-Pitch-Yaw representation, Exponential maps representation, and Quaternions representation.

- **Euler-angle representation** specifies a rotation by three successive rotations with three Euler angles (α, β, γ). Three rotations correspond to a sequence of $Z-Y-Z$ with respect to its current own axis. That is, first rotate about the z axis by the angle α, a second rotation about the new y axis by the angle β and a final rotation about the new z axis by the angle γ. Therefore, a successive postmultiplication of basic transformations in Eq. B.1 is needed,

$$\boldsymbol{R} = Rot(z,\alpha)Rot(y,\beta)Rot(z,\gamma). \tag{B.5}$$

  where $\boldsymbol{R}$ is extended to 4×4 for the consistency of matrix multiplication.

- **Roll-Pitch-Yaw representation** adopts a similar three successive rotation

with angles (α, β, γ) to represent a rotation. Different from the Euler method, these three rotations are specified as a sequence of $X - Y - Z$ with respect to a fixed coordinate frame. Therefore, a successive premultiplication of basic transformations in Eq. B.1 is desired,

$$\boldsymbol{R} = Rot(z, \gamma)Rot(y, \beta)Rot(x, \alpha). \tag{B.6}$$

- **Exponential maps representation** parameterizes a rotation by two items: a unit vector $\boldsymbol{r} = [r_x, r_y, r_z]^\top$ indicating the direction of a directed axis, and an angle θ describing the angle of the rotation about the axis. This is also referred as "Axis-angle representation". The exponential map is used as a transformation from axis-angle representation of rotations to rotation matrices by:

$$R = \exp([\boldsymbol{r}]_\times \theta) = \sum_{k=0}^{\infty} \frac{([\boldsymbol{r}]_\times \theta)^k}{k!} = I + [\boldsymbol{r}]_\times \theta + \frac{1}{2}([\boldsymbol{r}]_\times \theta)^2 + \frac{1}{6}([\boldsymbol{r}]_\times \theta)^3 + \cdots \tag{B.7}$$

where $[\boldsymbol{r}]_\times$ is the screw-symmetric matrix derived from the vector $\boldsymbol{r}$ by:

$$[\boldsymbol{r}]_\times = \begin{bmatrix} 0 & -r_z & -r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}. \tag{B.8}$$

With the Rodrigues's rotation formula, Eq. B.7 can be converted into

$$R = \boldsymbol{I} + [\boldsymbol{r}]_\times \sin\theta + [\boldsymbol{r}]_\times^2 (1 - \cos\theta). \tag{B.9}$$

- **Quaternions representation** adopts a quaternion form to express the axis-

angle representation $(\boldsymbol{r}, \theta)$ of a rotation by

$$\boldsymbol{q} = [cos\frac{\theta}{2} \;\; r_x sin\frac{\theta}{2} \;\; r_y sin\frac{\theta}{2} \;\; r_z sin\frac{\theta}{2}]^\top. \tag{B.10}$$

To extracting the three Euler angles or Roll-Yaw-Pitch angles from a given rotation matrix can be easily realized by obtaining their analytical expression. Though these two strategies can do a good job in a large range of camera orientations, they have a common drawback: When two of the three rotations axes align, one rotation has no effect. This problem is known as gimbal lock with infinitely solutions can be found in this case.

The quaternion representation avoids gimbal lock but estimation techniques must contain a constrained optimization approach to ensure the norm of $\boldsymbol{q}$ to remain one, which increases the algorithm complexity and is therefore not desirable in general. The exponential map represents a rotation as a 3-vector that gives its axis and magnitude. It avoids the gimbal lock problem and does not require an additional constraint to preserve the norm of a quaternion. Consequently, it is a very appropriate formulation to represent a rotation.

Appendix C

Forward Kinematics

Robot kinematics is to determine the position and orientation of the end effector given the values of the joint variables of a robot. It is concerned with establishing various coordinate systems and transformations among these coordinate systems. A typical diagram of a robot system is shown in Figure C.1 on page 166. The frames $\sum_o$, $\sum_b$, $\sum_c$, $\sum_e$ are assigned in the object, robot base, camera and end-effector respectively.

Denavit-Hartenberg convention, or DH convention is a commonly used convention for setting coordinate frames in robotic applications. In this convention, each joint of the robot is assigned a coordinate frame (Figure C.2 on page 167). The homogeneous transformation ${}^{i}\boldsymbol{T}_{i+1}$ from frame $\sum_i$ on joint i to the next frame $\sum_{i+1}$ on join $i+1$ is represented as a product of four basic transformations:

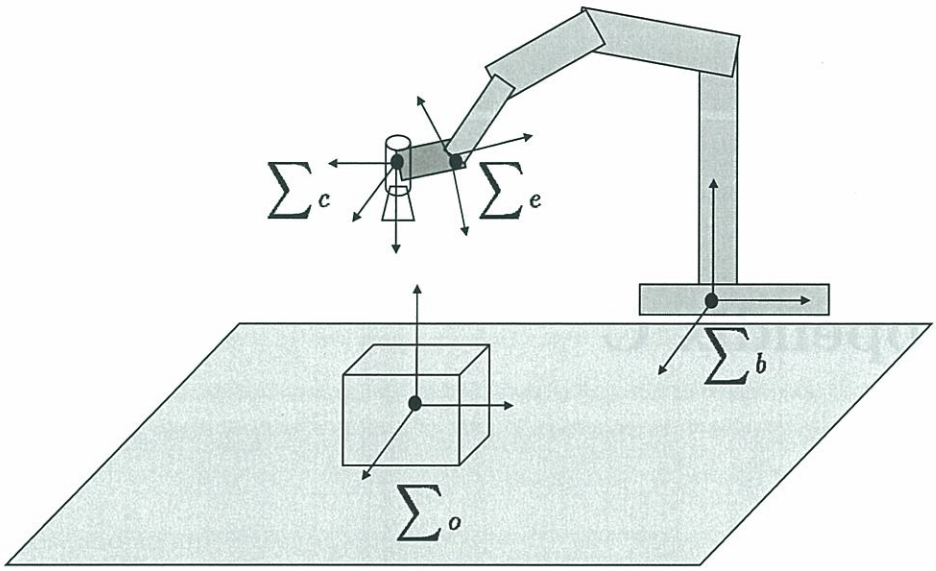


Figure C.1: Various coordinate frames in a typical robot system.

$$
\begin{aligned}
{}^{i}\boldsymbol{T}_{i+1} &= Rot(z,\theta_i)Trans(z,d_i)Trans(x,a_i)Rot(x,\alpha_i) \\
&= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}
\tag{C.1}
$$

where the four quantities θ_i, a_i, d_i, α_i are parameters associated with link i and joint i. The four parameters θ_i, a_i, d_i, α_i are generally named as *joint angle, link length, link offset*, and *link twist* respectively.

Our robot applications are implemented with a six degrees of freedom (DOF) "EPSON Pro Six PS5" robot, which is composed of six revolute joints with joint variables θ_0, θ_1, θ_2, θ_3, θ_4, θ_5. The DH parameters are listed in Table C.1. With these parameters, the pose of the end-effector ${}^{0}\boldsymbol{T}_{6}$ with respect to the robot base

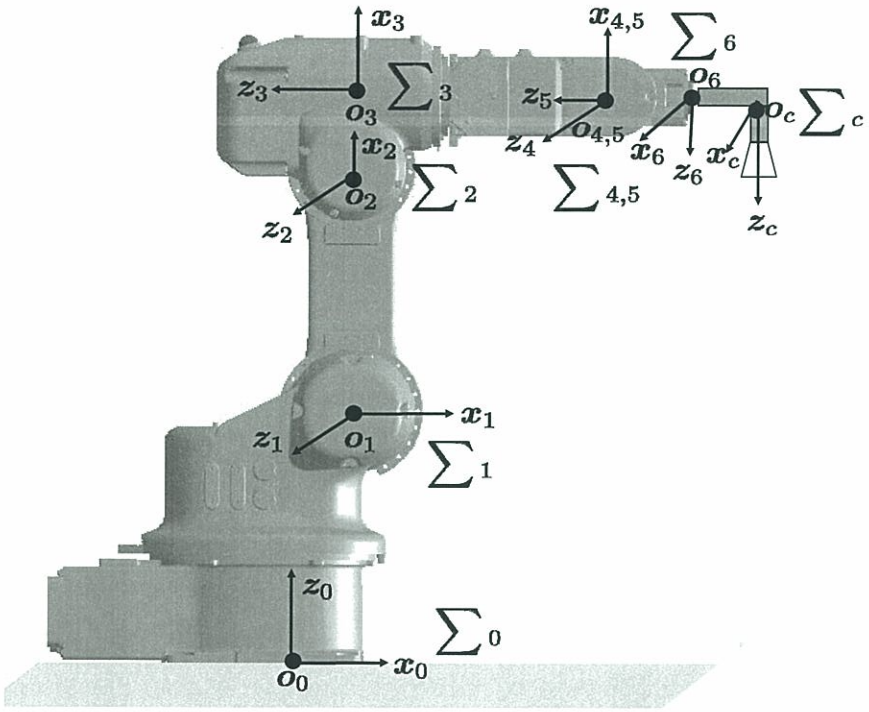


Figure C.2: Six coordinate frames $\sum_1$ to $\sum_6$ have been assigned with respect to the six joints of the robot "EPSON Pro Six PS5" with DH convention. $\sum_0$ refers to the robot base frame ($\sum_b$), while $\sum_c$ is the camera frame.

Table C.1: DH parameters for six-DOF robot "EPSON Pro Six PS5".

Link	θ_i	a_i (m)	d_i (m)	α_i
1	θ_0	0.100	0.300	90°
2	θ_1	0.290	0	0°
3	θ_2	0.085	0	−90°
4	θ_3	0	−0.300	90°
5	θ_4	0	0	−90°
6	θ_5	0	−0.090	0°

frame $\sum_0$ is expressed as a product of the six transformations

$$ {}^0\boldsymbol{T}_6 = \begin{bmatrix} {}^0\boldsymbol{R}_6 & {}^0\boldsymbol{t}_6 \\ 0 & 1 \end{bmatrix} = {}^0\boldsymbol{T}_1\ {}^1\boldsymbol{T}_2\ {}^2\boldsymbol{T}_3\ {}^3\boldsymbol{T}_4\ {}^4\boldsymbol{T}_5\ {}^5\boldsymbol{T}_6. \tag{C.2} $$

Appendix D

Velocity Transformation

In this part we discuss the relationships of object velocities among various coordinate frames in the robot application. We take the six DOF robot "EPSON Pro Six PS5" as an example. The output of the visual servo is the velocity screw with respect to the camera frame $\sum_c$, that is ${}^c\boldsymbol{V}_{6\times1} = [\boldsymbol{v}\ \boldsymbol{\omega}]^\top$. Therefore, the transformation from camera frame $\sum_c$ to the robot base frame $\sum_0$ in Figure C.2 is required to move the robot to fulfill such camera velocity demand. As the camera is fixed on the end effector, the relationship between the camera velocity and the end-effector velocity is discussed first. If the camera frame is coincident with the end-effector frame, the end-effector velocity will be the same as the camera velocity. However, in most applications, the camera frame is not coincident with the end-effector frame, but is rigidly attached to it. Suppose the transformation from the end-effector frame $\sum_6$ (the end-effector frame is assigned on the sixth joint) to the camera frame $\sum_c$ is

$$ {}^6\boldsymbol{T}_c = \begin{bmatrix} {}^6\boldsymbol{R}_c & {}^6\boldsymbol{t}_c \\ 0 & 1 \end{bmatrix}. \tag{D.1} $$

Then the velocity expressed in the end-effector frame ${}^{6}\boldsymbol{V}$ can be expressed as:

$$ {}^{6}\boldsymbol{V} = \begin{bmatrix} {}^{6}\boldsymbol{R}_c & [{}^{6}\boldsymbol{t}_c]_{\times}\, {}^{6}\boldsymbol{R}_c \\ \mathbf{0} & {}^{6}\boldsymbol{R}_c \end{bmatrix} {}^{c}\boldsymbol{V}, \tag{D.2} $$

where $[{}^{6}\boldsymbol{t}_c]_{\times}$ is the screw-symmetric matrix derived from the 3×1 vector ${}^{6}\boldsymbol{t}_c$ as expressed in Eq. B.8.

After obtaining the end-effector velocity ${}^{6}\boldsymbol{V}$ in the end-effector frame, a simple transformation is adopted to get this velocity expressed in the robot base frame $\sum_0$:

$$ {}^{0}\boldsymbol{V} = \begin{bmatrix} {}^{0}\boldsymbol{R}_6 & \mathbf{0} \\ \mathbf{0} & {}^{0}\boldsymbol{R}_6 \end{bmatrix} {}^{6}\boldsymbol{V}. \tag{D.3} $$

Given this velocity ${}^{0}\boldsymbol{V}$, the robot Jacobian matrix $\boldsymbol{J}$ is used to determine the desired joint velocities $\dot{\boldsymbol{\theta}} = [\dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5]$ of the robot. The Jacobian matrix specifies the relationship between end-effector velocity ${}^{0}\boldsymbol{V}$ and the joints velocity $\dot{\boldsymbol{\theta}}$ by

$$ {}^{0}\boldsymbol{V} = \boldsymbol{J}\dot{\boldsymbol{\theta}}. \tag{D.4} $$

From Eq. D.4, the inverse velocity problem of finding the joint velocities $\dot{\boldsymbol{\theta}}$ that produce the desired end-effector velocity ${}^{0}\boldsymbol{V}$ can be solved by simply inverting the Jacobian matrix

$$ \dot{\boldsymbol{\theta}} = \boldsymbol{J}^{-1}\, {}^{0}\boldsymbol{V}. \tag{D.5} $$

The Jabobian matrix can be derived from the velocity transformations between

each successive joint frame $\sum_i$ and $\sum_{i+1}$,

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{v1} & \boldsymbol{J}_{v2} & \boldsymbol{J}_{v3} & \boldsymbol{J}_{v4} & \boldsymbol{J}_{v5} & \boldsymbol{J}_{v6} \\ \boldsymbol{J}_{\omega 1} & \boldsymbol{J}_{\omega 2} & \boldsymbol{J}_{\omega 3} & \boldsymbol{J}_{\omega 4} & \boldsymbol{J}_{\omega 5} & \boldsymbol{J}_{\omega 6} \end{bmatrix}_{6\times 6}, \tag{D.6}$$

where the linear velocity Jacobian $\boldsymbol{J}_{vi}$ and angular velocity Jacobian $\boldsymbol{J}_{\omega i}$ are defined as follows:

$$\begin{aligned} \boldsymbol{J}_{vi} &= \boldsymbol{z}_{i-1} \times (\boldsymbol{o}_i - \boldsymbol{o}_{i-1}), \\ \boldsymbol{J}_{\omega i} &= \boldsymbol{z}_{i-1}. \end{aligned} \tag{D.7}$$

The $\boldsymbol{z}_i$ and $\boldsymbol{o}_i$ are the vector representations of $\boldsymbol{z}$ axis and origin point of the ith joint coordinate frame $\sum_i$.

Bibliography

[1] K. Hashimoto, Ed., *Visual Servoing Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, World Scientific Publishing, 1993.

[2] S. Hutchinson, *et al.*, "A tutorial on visual servo control", *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, 1996.

[3] C. Samson, *et al.*, *Robot Control: The Task Function Approach*, Oxford University Press, 1991.

[4] F. Chaumette and S. Hutchinson, "Visual servo control. I. basic approaches", *IEEE Robot. Automat. Mag.*, vol. 13, no. 4, pp. 82–90, 2006.

[5] F. Chaumette and S. Hutchinson, "Visual servo control. II. advanced approaches", *IEEE Robot. Automat. Mag.*, vol. 14, no. 1, pp. 109–118, 2007.

[6] L. Weiss, *et al.*, "Dynamic sensor-based control of robots with visual feedback", *IEEE Journal of Robotics and Automation*, vol. 3, no. 5, pp. 404–417, 1987.

[7] W. J. Wilson, *et al.*, "Relative end-effector control using Cartesian position based visual servoing", *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 684–696, 1996.

[8] B. Ronen, *et al.*, "Visual homing: surfing on the epipoles", *International Journal of Computer Vision*, vol. 33, no. 2, pp. 117–137, 1999.

[9] C. J. Taylor and J. P. Ostrowski, "Robust vision-based pose control", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2000)*, vol. 3, pp. 2734–2740, 2000.

[10] B. Thuilot, *et al.*, "Position based visual servoing: keeping the object in the field of vision", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2002)*, vol. 2, pp. 1624–1629, 2002.

[11] E. Malis and F. Chaumette, "Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods", *IEEE Trans. Robot. Autom.*, vol. 18, no. 2, pp. 176–186, 2002.

[12] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections", In *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, Martin A. Fischler and Oscar Firschein, Eds., pp. 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[13] R. I. Hartley, "Estimation of relative camera positions for uncalibrated cameras", In *Proc. 2nd European Conf. on Computer Vision*, ECCV 1992, pp. 579–587, London, UK, 1992. Springer-Verlag.

[14] O. Faugeras, *Three-dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, Cambridge, MA, USA, 1993.

[15] Y. Ma, *et al.*, *An Invitation to 3-D Vision: From Images to Geometric Models*, SpringerVerlag, 2003.

[16] P. Rives, "Visual servoing based on epipolar geometry", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2000)*, vol. 1, pp. 602–607, 2000.

[17] O. Faugeras, *et al.*, *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*, MIT Press, Cambridge, MA, USA, 2001.

[18] E. Malis and F. Chaumette, "2 1/2 D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement", *International Journal of Computer Vision*, vol. 37, no. 1, pp. 79–97, 2000.

[19] G. Chesi, *et al.*, "Keeping features in the field of view in eye-in-hand visual servoing: a switching approach", *IEEE Trans. Robot.*, vol. 20, no. 5, pp. 908–914, 2004.

[20] M. Iwatsuki and N. Okiyama, "A new formulation of visual servoing based on cylindrical coordinate system", *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 266–273, 2005.

[21] B. Espiau, *et al.*, "A new approach to visual servoing in robotics", *IEEE Trans. Robot. Autom.*, vol. 8, no. 3, pp. 313–326, 1992.

[22] F. Chaumette, *et al.*, "Classification and realization of the different vision-based tasks", In *Visual Servoing, Robotics and Automated Systems*, K. Hashimoto, Ed., vol. 7, pp. 199–228. World Scientific, Singapore, 1993.

[23] H. Wang, *et al.*, "Uncalibrated dynamic visual servoing using line features", In

Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2008), pp. 3046–3051, 2008.

[24] F. Chaumette, "Image moments: a general and useful set of features for visual servoing", *IEEE Trans. Robot.*, vol. 20, no. 4, pp. 713–723, 2004.

[25] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects", *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1116–1127, 2005.

[26] R. T. Fomena and F. Chaumette, "Using spherical moments for visual servoing from a special target with unique projection center cameras", In *Proc. 10th Int. Conf. Control, Automation, Robotics and Vision (ICARCV 2008)*, pp. 119–124, 2008.

[27] C. Copot, *et al.*, "Image moments based visual control algorithm for servoing systems", In *Proc. IEEE 5th Int. Conf. Intelligent Computer Communication and Processing (ICCP 2009)*, pp. 157–160, 2009.

[28] G. Li, *et al.*, "Visual servoing using image moments and nonlinear state error feedback control law", In *Proc. Chinese Control and Decision Conf. (CCDC)*, pp. 3209–3214, 2010.

[29] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true Jacobian", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 1994)*, vol. 1, pp. 186–193, 1994.

[30] M. Hao, *et al.*, "Uncalibrated eye-in-hand visual servoing using recursive least

squares", In *Proc. ISIC IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 64–69, 2007.

[31] M. Jagersand, *et al.*, "Experimental evaluation of uncalibrated visual servoing for precision manipulation", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1997)*, vol. 4, pp. 2874–2880, 1997.

[32] K. Deguchi, "Direct interpretation of dynamic images and camera motion for visual servoing without image feature correspondence", *Journal of Robotics and Mechatronics*, vol. 9, no. 2, pp. 104–110, 1997.

[33] F. Chaumette, "Potential problems of stability and convergence in Image-based and Position-based visual servoing", In *LNCIS Series In the Confluence of Vision and Control*, vol. 237, pp. 66–78. Springer Verlag, 1998.

[34] N. R. Gans and S. A. Hutchinson, "An asymptotically stable switched system visual controller for eye in hand robots", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2003)*, vol. 1, pp. 735–742, 2003.

[35] N. R. Gans and S. A. Hutchinson, "Stable visual servoing through hybrid switched-system control", *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 530–540, 2007.

[36] A. H. A. Hafez and C. V. Jawahar, "Visual servoing by optimization of a 2D/3D hybrid objective function", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2007)*, pp. 1691–1696, 2007.

[37] A. H. A. Hafez, *et al.*, "Hybrid visual servoing by boosting IBVS and PBVS",

In *Proc. 3rd Int. Conf. Information and Communication Technologies: From Theory to Applications (ICTTA 2008)*, pp. 1–6, 2008.

[38] O. Kermorgant and F. Chaumette, "Combining IBVS and PBVS to ensure the visibility constraint", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2011)*, pp. 2849–2854, 2011.

[39] E. Malis, et al., "2 1/2 D visual servoing", *IEEE Trans. Robot. Autom.*, vol. 15, pp. 238–250, 1999.

[40] F. Chaumette and E. Malis, "2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2000)*, vol. 1, pp. 630–635, 2000.

[41] M. Marey and F. Chaumette, "New strategies for avoiding robot joint limits: Application to visual servoing using a large projection operator", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2010)*, pp. 6222–6227, 2010.

[42] O. D. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

[43] M. Vargas and E. Malis, "Visual servoing based on an analytical homography decomposition", In *Proc. 44th IEEE Conf. Decision and Control and 2005 European Control (CDC-ECC 2005)*, pp. 5379–5384, 2005.

[44] E. Malis and M. Vargas, "Deeper understanding of the homography decomposi-

tion for vision-based control", Technical report, Research Report 6303, INRIA, 2007.

[45] Y. Fang, *et al.*, "Homography-based visual servoing of wheeled mobile robots", In *Proc. 41st IEEE Conf. Decision and Control*, vol. 3, pp. 2866–2871, 2002.

[46] J. Chen, *et al.*, "Homography-based visual servo tracking control of a wheeled mobile robot", *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 406–415, 2006.

[47] S. Benhimane and E. Malis, "Homography-based 2D Visual Tracking and Servoing", *International Journal of Robotic Research*, vol. 26, pp. 661–676, 2007.

[48] G. Silveira and E. Malis, "Direct visual servoing with respect to rigid objects", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2007)*, pp. 1963–1968, 2007.

[49] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 1998)*, vol. 2, pp. 705–711, 1998.

[50] G. Hu, *et al.*, "A quaternion formulation for homography-based visual servo control", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2006)*, pp. 2391–2396, 2006.

[51] G. Hu, *et al.*, "Adaptive homography-based visual servo tracking control via a quaternion formulation", *IEEE Trans. Control Syst. Techno.*, vol. vol. 18, no. 1, pp. 128–135, 2010.

[52] T. Koenig and G. N. D. Souza, "Implementation of a Homography-based Visual

Servo Control using a Quaternion Formulation.", In *ICINCO-RA (1)*, pp. 288–294. INSTICC Press, 2008.

[53] D. S. Kumar and C. V. Jawahar, "Robust homography-based control for camera positioning in piecewise planar environments", In *Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, pp. 906–918, 2006.

[54] R. M. Haralick, *et al.*, "Pose estimation from corresponding point data", *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 6, pp. 1426–1446, 1989.

[55] G. Pierre and A. K. Omar, "Solutions for exterior orientation in photogrammetry: A review", *The Photogrammetric Record*, vol. 17, pp. 615–634, 2002.

[56] Z. Liu, *et al.*, "Stereo-based head pose tracking with motion compensation model", In *Proc. IEEE Int. Conf. Robotics and Biomimetics (ROBIO 2004)*, pp. 700–704, 2004.

[57] R. Laganiere, *et al.*, "Robust object pose estimation from feature-based stereo", *IEEE Trans. Instrum. Meas.*, vol. 55, no. 4, pp. 1270–1280, 2006.

[58] G. Spampinato, *et al.*, "An embedded stereo vision module for 6D pose estimation and mapping", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2011)*, pp. 1626–1631, 2011.

[59] J. H. Kim, *et al.*, "Visual measurement of a 3-D plane pose by a cylindrical structured light", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 1993)*, vol. 3, pp. 1845–1850, 1993.

[60] Meng. Y. and H. Zhuang, "Reliable solution for object pose determination using

an active vision system", In *Proc. Int. Conf. Advanced Robotics (ICAR 2005)*, pp. 392–397, 2005.

[61] B. Zhang and Y. F. Li, "A study on the relative pose problem in an active vision system with varying focal lengths", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2007)*, pp. 1092–1097, 2007.

[62] J. K. Oh, *et al.*, "Development of a structured-light sensor based bin-picking system using ICP algorithm", In *Proc. Int. Conf. Control Automation and Systems (ICCAS 2010)*, pp. 1673–1677, 2010.

[63] F. Prieto, *et al.*, "Visual system for the fast and automated inspection of 3D parts", *International Journal of CAD/CAM and Computer Graphic*, vol. 13, no. 4, pp. 211–227, 1998.

[64] A. Dubrawski and I. Siemiatkowska, "A method for tracking the pose of a mobile robot equipped with a scanning laser range finder", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1998)*, vol. 3, pp. 2518–2523, 1998.

[65] P. Jensfelt and H. I. Christensen, "Laser based pose tracking", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1999)*, vol. 4, pp. 2994–3000, 1999.

[66] A. C. Victorino and P. Rives, "A relative motion estimation by combining laser measurement and sensor based control", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2002)*, vol. 4, pp. 3924–3929, 2002.

[67] J. A. Hesch, *et al.*, "A 3D pose estimator for the visually impaired", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2009)*, pp. 2716–2723, 2009.

[68] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", In *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, Martin A. Fischler and Oscar Firschein, Eds., pp. 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[69] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[70] A. A. Transeth, et al., "A robotic concept for remote maintenance operations: A robust 3D object detection and pose estimation method and a novel robot tool", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5099–5106, 2010.

[71] I. H. Richard and S. Peter, "Triangulation", *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[72] C. Tomasi, "Shape and motion from image streams under orthography: a factorization method", *International Journal of Computer Vision*, vol. 9, pp. 137–154, 1992.

[73] D. D. Morris and T. Kanade, "A unified factorization algorithm for points, line segments and planes with uncertainty models", In *Proc. 6th Int. Conf. Computer Vision*, pp. 696–702, 1998.

[74] C. J. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 3, pp. 206–218, 1997.

[75] S. Lehmann, *et al.*, "Correspondence-free determination of the affine fundamental matrix", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 82–97, 2007.

[76] A. Makadia, *et al.*, "Correspondence-free structure from motion", *International Journal of Computer Vision*, vol. 75, pp. 311–327, 2007.

[77] F. Dellaert, *et al.*, "Structure from motion without correspondence", In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 557–564, 2000.

[78] W. Lin,*et al.*, "Simultaneous camera pose and correspondence estimation in cornerless images", In *Proc. IEEE 12th Int. Conf. Computer Vision*, pp. 1179–1186, 2009.

[79] W. Lin, *et al.*, "Simultaneous camera pose and correspondence estimation with motion coherence", *International Journal of Computer Vision*, vol. 93, no. 1, pp. 1–17, 2011.

[80] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", In *Proc. Int. Joint Conf. on Artificial Intelligence*, pp. 674–679, 1981.

[81] E. Malis, "Improving vision-based control using efficient second-order minimization techniques", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2004)*, vol. 2, pp. 1843–1848, 2004.

[82] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a

video-based augmented reality conferencing system", In *Proc. 2nd IEEE and ACM Int. Augmented Reality Workshop (IWAR '99)*, pp. 85–94, 1999.

[83] V. Lepetit and P. Fua, "Monocular model-based 3D tracking of rigid objects: a survey", In *Foundations and Trends in Computer Graphics and Vision*, pp. 1–89, 2005.

[84] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2nd edition, 2003.

[85] F. D. Daniel and S. D. Larry, "Model-based object pose in 25 lines of code", *International Journal of Computer Vision*, vol. 15, pp. 123–141, 1995.

[86] X. Ying and H. Zha, "Camera pose determination from a single view of parallel lines", In *Proc. Int. Conf. Image Processing (ICIP 2005)*, vol. 3, pp. 1056–1059, 2005.

[87] F. Ababsa, "Robust Extended Kalman Filtering for camera pose tracking using 2D to 3D lines correspondences", In *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics (AIM 2009)*, pp. 1834–1838, 2009.

[88] S. Li and Y. Hai, "Estimating camera pose from H-pattern of parking lot", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2010)*, pp. 3954–3959, 2010.

[89] F. Ababsa and M. Mallem, "Robust circular fiducials tracking and camera pose estimation using particle filtering", In *Proc. ISIC IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 1159–1164, 2007.

[90] S. Nagarajan, *et al.*, "Determining camera pose using free-form lines", In *Proc. Western New York Image Processing Workshop (WNYIPW)*, pp. 50–53, 2010.

[91] F. Shi and Y. Liu, "Estimation of camera pose using 2D to 3D corner correspondence", In *Proc. Int. Conf. Information Technology: Coding and Computing (ITCC 2004)*, vol. 2, pp. 805–809, 2004.

[92] P. Bouthemy, "A maximum likelihood framework for determining moving edges", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 5, pp. 499–511, 1989.

[93] C. Harris, *Tracking with Rigid Models*, MIT Press, Cambridge, MA, USA, 1993.

[94] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 932–946, 2002.

[95] A. I. Comport, *et al.*, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework", *IEEE Trans. Vis. Comput. Graphics*, vol. 12, no. 4, pp. 615–628, 2006.

[96] P. David, *et al.*, "SoftPOSIT: Simultaneous pose and correspondence determination.", In *ECCV (3)*, vol. 2352 of *Lecture Notes in Computer Science*, pp. 698–714. Springer, 2002.

[97] S. Gold, *et al.*, "New algorithms for 2D and 3D point matching: pose estimation and correspondence", *Pattern Recognition*, vol. 31, no. 8, pp. 1019–1031, 1998.

[98] P. David, *et al.*, "Simultaneous pose and correspondence determination using

line features", In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 424–431, 2003.

[99] C. Harris and M. Stephens, "A combined corner and edge detector", In *Proc. 4th Alvey Vision Conf.*, pp. 147–151, 1988.

[100] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, pp. 91–110, November 2004.

[101] J. M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison", *SIAM Journal on Imaging Sciences*, vol. 2, pp. 438–469, 2009.

[102] K. Yan and R. Sukthankar, "PCA-SIFT: a more distinctive representation for local image descriptors", In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR 2004)*, vol. 2, pp. 506–513, 2004.

[103] S. Warn, *et al.*, "Accelerating SIFT on parallel architectures", In *Proc. IEEE Int. Conf. Cluster Computing and Workshops (CLUSTER 2009)*, pp. 1–4, 2009.

[104] J. Zhang, *et al.*, "Overview of approaches for accelerating scale invariant feature detection algorithm", In *Proc. Int. Conf. Electric Information and Control Engineering (ICEICE 2011)*, pp. 585–589, 2011.

[105] B. Herbert, *et al.*, "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.

[106] M. A. R. Ahad, *et al.*, "SURF-based spatio-temporal history image method for action representation", In *Proc. Int. Conf. Industrial Technology (ICIT 2011)*, pp. 411–416, 2011.

[107] M. Calonder, *et al.*, "BRIEF: Binary Robust Independent Elementary Features", In *European Conf. Computer Vision*, vol. 4, pp. 778–792, 2010.

[108] N. Cornelis and G. L. Van, "Fast scale invariant feature detection and matching on programmable graphics hardware", In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition Workshops (CVPRW 2008)*, pp. 1–8, 2008.

[109] J. Svab, *et al.*, "FPGA based speeded up robust features", In *Proc. IEEE Int. Conf. Technologies for Practical Robot Applications (TePRA 2009)*, pp. 35–41, 2009.

[110] E. Tola, *et al.*, "DAISY: An efficient dense descriptor applied to wide-baseline stereo", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, 2010.

[111] J. Fischer, *et al.*, "A rotation invariant feature descriptor O-DAISY and its FPGA implementation", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2011)*, pp. 2365–2370, 2011.

[112] W. E. L. Grimson and D. P. Huttenlocher, "On the verification of hypothesized matches in model-based recognition", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 12, pp. 1201–1213, 1991.

[113] J. H. M. Byne and J. A. D. W. Anderson, "A CAD-based computer vision system", *Image and Vision Computing*, vol. 16, no. 8, pp. 533–539, June 1998.

[114] H. Borotschnig, *et al.*, "Appearance based active object recognition", *Inter-*

national Journal of Image and Vision Computing, vol. 18, no. 9, pp. 715–728, 2000.

[115] C. M. Cyr and B. B. Kimia, "3D object recognition using shape similiarity-based aspect graph", In *Proc. 8th IEEE Int. Conf. Computer Vision (ICCV 2001)*, vol. 1, pp. 254–261, 2001.

[116] T. Su, *et al.*, "Shape memorization and recognition of 3D objects using a similarity-based aspect-graph approach", In *Proc. IEEE Int. Conf. Systems, Man and Cybernetics (SMC 2006)*, vol. 6, pp. 4920–4925, 2006.

[117] M. Ulrich, *et al.*, "CAD-based recognition of 3D objects in monocular images", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2009)*, pp. 1191–1198, 2009.

[118] F. Ababsa and M. Mallem, "Robust camera pose estimation combining 2D/3D points and lines tracking", In *Proc. IEEE Int. Symp. Industrial Electronics (ISIE 2008)*, pp. 774–779, 2008.

[119] L. Masson, *et al.*, "Contour/texture approach for visual tracking", In *Proc. 13th Scandinavian Conf. Image Analysis*, pp. 661–668. Springer-Verlag, 2003.

[120] M. Pressigout and E. Marchand, "Real-time hybrid tracking using edge and texture information", *International Journal of Robotics Research*, vol. 26, pp. 689–713, July 2007.

[121] A. Ladikos, *et al.*, "A real-time tracking system combining template-based and feature-based approaches", In *Proc. Intl. Conf. on Computer Vision Theory and Applications*, vol. 2, pp. 325–332, Barcelona, Spain, 2007.

[122] C. Choi and H. I. Christensen, "Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2010)*, pp. 4048–4055, 2010.

[123] B. Barrois, et al., "3D pose estimation of vehicles using a stereo camera", In *Proc. IEEE Intelligent Vehicles Symp.*, pp. 267–272, 2009.

[124] M. Liu, *et al.*, "Pose estimation in heavy clutter using a multi-flash camera", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2010)*, pp. 2028–2035, 2010.

[125] H. S. Hong and M. J. Chung, "3D pose and camera parameter tracking algorithm based on Lucas-Kanade image alignment algorithm", In *Proc. Int. Conf. Control, Automation and Systems (ICCAS 2007)*, pp. 548–551, 2007.

[126] T. Inoue and K. Hashimoto, *Recognition and pose estimation of industrial component*, Master thesis, Tohoku University, Japan, 2012.

[127] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[128] H.Y. Shum and R. Szeliski, "Systems and experiment paper: construction of panoramic image mosaics with global and local alignment", *International Journal of Computer Vision*, vol. 36, no. 2, pp. 101–130, February 2000.

[129] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2004)*, vol. 1, pp. 943–948, 2004.

[130] E. Ito, *et al.*, "GPU-based high-speed and high-precision visual tracking", In *Proc. 2010 SICE Annu. Conf.*, pp. 2151–2154, 2010.

[131] E. Malis, "Improving vision-based control using efficient second-order minimization techniques", In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2004)*, vol. 2, pp. 1843–1848, 2004.

[132] E. Malis, "An efficient unified approach to direct visual tracking of rigid and deformable surfaces", In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2007)*, pp. 2729–2734, 2007.

[133] J. Shimamura, *et al.*, "Multiple-plane detection based on homography and projected line detection for use in a projector-camera system", *Journal of The Institute of Image Information and Television Engineers*, vol. 61, no. 1, pp. 76–84, 2007.

[134] D. G. Lowe, "Object recognition from local scale-invariant features", In *Proc. 7th IEEE Int. Conf. Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[135] S. Heymann, *et al.*, "SIFT implementation and optimization for general-purpose gpu", In *WSCG2007*, 2007.

[136] Q. Zhang, *et al.*, "SIFT implementation and optimization for multi-core systems", In *Proc. IEEE Int. Symp. Parallel and Distributed Processing (IPDPS 2008)*, pp. 1–8, 2008.

[137] C. Wu, *et al.*, "Detecting large repetitive structures with salient boundaries", In *Proc. 11th European Conf. Computer vision: Part II*, pp. 142–155, Berlin, Heidelberg, 2010.

[138] P. Viola and W. M. III. Wells, "Alignment by maximization of mutual information", In *Proc. 5th Int. Conf. on Computer Vision*, pp. 16–23, 1995.

[139] A. Dame, *A unified direct approach for visual servoing and visual tracking using mutual information*, Ph.D. dissertation, University of Rennes, France, 2011.

[140] M. Li, "Correspondence analysis between the image formation pipelines of graphics and vision", In *Proc. 9th Spanish Symp. Pattern Recognition and Image Analysis*, pp. 187–192, 2001.

List of Author's Publications

<Journal>

[1] C. Zang and K. Hashimoto, "GPU acceleration in a visual servo system", in *Journal of Robotics and Mechatronics*, vol. 24, no. 1, pp. 105-114, 2012.

<Conference-Full Paper Refereed>

[1] C. Zang and K. Hashimoto, "A flexible visual inspection system combining pose estimation and visual servo approaches", in *Proc. 2012 IEEE Int. Conf. Robotics and Automation (ICRA 2012)*, Minnesota, USA, 2012. (To appear).

[2] C. Zang and K. Hashimoto, "Camera localization by CAD model matching", in *Proc. 2011 IEEE/SICE Int. Symp. System Integration (SI International 2011)*, Kyoto, Japan, pp. 30-35, 2011.

[3] C. Zang et al., "A visual tracking strategy using Computer Graphics and edge", in *Proc. 2011 IEEE Int. Conf. Robotics and Biomimetics (IEEE-ROBIO 2011)*, Thailand, pp. 981-986, 2011.

[4] C. Zang and K. Hashimoto, "An improved visual inspection system using visual servo", in *Proc. 2010 IEEE/SICE Int. Symp. System Integration (SI International 2010)*, Sendai, Japan, pp. 13-18, 2010.

[5] C. Zang and K. Hashimoto, "A new approach of GPU accelerated visual tracking", in *LNCS 6475: Proc. 12th Int. Conf. Advanced Concepts for Intelligent Vision Systems (ACIVS 2010)*, Sydney, Australia, pp. 354-365, 2010.

[6] C. Zang and K. Hashimoto, "Using GPUs to improve system performance in visual servo systems", in *Proc. 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2010)*, Taipei, Taiwan, pp. 3937-3942, 2010.

[7] C. Zang and K. Hashimoto, "GPU acceleration in a visual servo system", in *Proc. 5th Int. Conf. Advanced Mechatronics (ICAM 2010)*, Osaka, Japan, pp. 7-12, 2010.

<Conference-Not Refereed>

[1] C. Zang et al, "GPU accelerating visual tracking", in *Proc. 2010 JSME Conf. Robotics and Mechatronics (ROBOMEC 2010)*, Asahigawa, Japan, 2010.

Acknowledgments

First of all, I would like to thank all the referees of this dissertation, Professor *Koichiro Deguchi*, Professor *Kazuhiro Kosuge*, Professor *Koich Hashimoto* and Associate Professor *Shingo Kagami* for their valuable comments and suggestions.

My special thanks go to my supervisor, Professor *Koichi Hashimoto*, who has led me into the wonderful world of computer vision and robotics, and is always encouraging me to explore new things in my research. I sincerely thank for his valuable suggestions on my research and great support with all my visit research programs. Professor*Koichi Hashimoto* has acted as a supervisor, a guide and also a friend to me in these three years. I will aways appreciate the honor of being a student of Professor *Koichi Hashimoto.*

I would like to thank Associate Professor *Shingo Kagami* for his advice and guidance throughout my whole study, especially for this dissertation. I am sincerely grateful to his beneficial advice and wise guidance.

I am grateful to Professor *Kazuhiro Kosuge* and Associate Professor *Yasuhisa Hirata* for valuable suggestions and comments in the collaborative research. I have enlarged my knowledge of robotics from this precious opportunity.

I would like to thank all the members in the Hashimoto-Kagami Lab and in the collaborative research. With their kindly help, I have spent a splendid three years on my Ph.D. Course in Tohoku University. Talking and working with them has greatly enhanced my understanding of Japanese culture. Working together and solving the problems in collaboration has impressed me so much and it will be an unforgettable experience in my life.

I would like to give special thanks to the secretary of our lab, *Mrs. Kaori Ichinohe*, who has helped me a lot to prepare the materials in the lab.

Special appreciation goes to Professor *Mitsuyuki Nakao* for accepting me into the ASIST program and offering me the financial support for my life in Japan. So that I can maintain a stable life and continue my Ph.D. study.

Sincerely thanks go to Dr. *Francois Chaumette* and Professor *Eric Merchand* for their kindly support and suggestions on my study when I stayed with the LAGADIC group in INRIA Rennes, France. The "ViSP" software platform by the LAGADIC group has helped me a lot to understand many complex problems in computer vision and visual servo.

Last but not least, I would like to express my great appreciation to my family. With their love, understanding and support, I can continue the academic life and realize this study.